Theoretical Computer Science 122 (1994) 225–262
Elsevier

On the duality of abduction and model generation in a framework for model generation with equality

Marc Denecker* and Danny De Schreye**

Department of Computer Science, K.U. Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium

Abstract

Denecker, M. and D. De Schreye, On the duality of abduction and model generation in a framework for model generation with equality, Theoretical Computer Science 122 (1994) 225–262.

We present a duality relationship between abduction for definite abductive programs and model generation on the only-if part of these programs. As was pointed out by Console et al. (1991), abductive solutions for an abductive program correspond to models of the only-if part. We extend this observation by showing that the procedural semantics of abduction itself can be interpreted dually as a form of model generation on the only-if part. This model generation extends Satchmo with an efficient treatment of equality atoms occurring in the head of rules. It is illustrated how this duality allows to improve current procedures for both abduction and model generation by transferring technical results known for one of these computational paradigms to the other.

1. Introduction

The work presented in this paper was motivated by some recent progress made in the field of logic programming to formalise abductive reasoning as logic deduction

Correspondence to: M. Denecker, Department of Computer Science, K.U. Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium. Email addresses: {marcd, dannyd}@cs.kuleuven.ac.be.

* Supported by the Belgian "Diensten voor Programmatie van Wetenschapsbeleid", under the contract RFO-AI-03.

** Supported by the Belgian National Fund for Scientific Research.

[4,2]. Kowalski [14] presents the intuition behind this approach. He considers the following simple definite abductive logic program:

$$P = \{wobbly\text{-}wheel \leftarrow flat\text{-}tyre$$
$$wobbly\text{-}wheel \leftarrow broken\text{-}spokes$$
$$flat\text{-}tyre \leftarrow punctured\text{-}tube$$
$$flat\text{-}tyre \leftarrow leaky\text{-}valve\},$$

where the predicates broken-spokes, punctured-tube and leaky-valve are the abducibles. Given a query $Q = \leftarrow wobbly\text{-}wheel$, abductive reasoning allows to infer the assumptions:

$$S_1 = \{punctured\text{-}tube\},$$
$$S_2 = \{leaky\text{-}valve\},$$
$$S_3 = \{broken\text{-}spokes\}.$$

These sets of assumptions are abductive solutions to the given query Q in the sense that, for each S_i, we have that $P \cup S_i \models \neg Q$.

Kowalski points out that we can equally well obtain these solutions by deduction if we first transform the abductive program $P \cup \{Q\}$ into a new logic theory T. The transformation consists in taking the only-if part of every definition of a nonabducible predicate in the Clark-completion of P and adding the negation of Q. In the example, we obtain the (non-Horn) theory T as follows:

$$T = \{wobbly\text{-}wheel \rightarrow flat\text{-}tyre, broken\text{-}spokes$$
$$flat\text{-}tyre \rightarrow punctured\text{-}tube, leaky\text{-}valve$$
$$wobbly\text{-}wheel \leftarrow \}.$$

Minimal models for this new theory T are:

$$M_1 = \{wobbly\text{-}wheel, flat\text{-}tyre, punctured\text{-}tube\},$$
$$M_2 = \{wobbly\text{-}wheel, flat\text{-}tyre, leaky\text{-}valve\},$$
$$M_3 = \{wobbly\text{-}wheel, broken\text{-}spokes\}.$$

Restricting these models to the atoms of the abducible predicates only, we precisely obtain the three abductive solutions S_1, S_2 and S_3 of the original problem.

The above observation points to an interesting issue; namely, the possibility of linking these dual declarative semantics by completely equivalent dual procedures. Figure 1 shows this duality between an SLD + abduction tree [6] and the execution tree of Satchmo, a theorem prover based on model generation [18].

Although this example illustrates the potential of using deduction or, more precisely, model generation, as a formalisation of abductive reasoning, an obvious

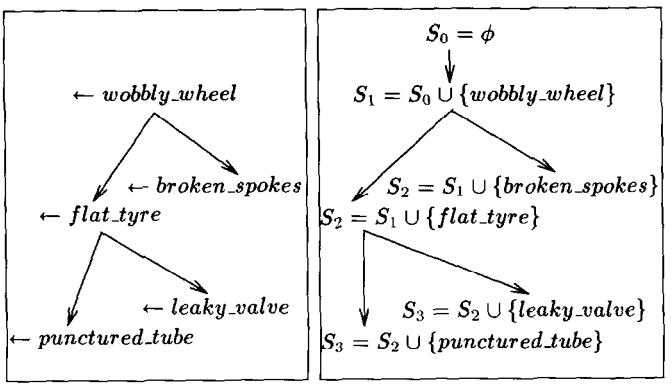

Fig. 1. Procedural duality of abduction and Satchmo.

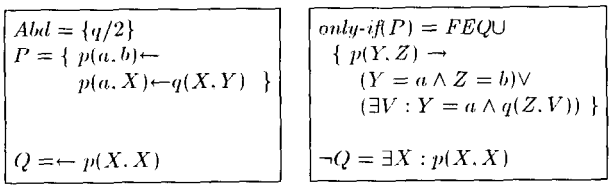

Fig. 2. A predicate example.

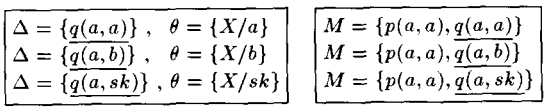

Fig. 3. Abductive solutions and models.

restriction of the example is that it is only propositional. Would this approach also hold for the general case of definite abductive programs? An example of a nonpropositional program and its only-if part is given in Fig. 2.

The theory *only-if*(P) consists not only of the only-if part of the definitions of the predicates but comprises also the axioms of free equality (FEQ), also known as Clark equality [3]. The abductive solutions and models of *only-if*(P) are displayed in Fig. 3.

The duals of the abductive solutions are again identical to models of *only-if*(P). This example suggests that at least the duality on the level of declarative semantics is maintained. However, on the level of procedural semantics, some difficulties arise. An SLD + abduction derivation tree is given in Fig. 4. After skolemisation of the residue $\leftarrow q(a, V)$, we obtain the third abductive solution.

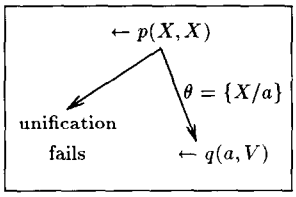

Fig. 4. Abductive derivation tree.

With respect to the model generation, a first problem is that *only-if* (*P*) is not clausal and Satchmo cannot deal with nonclausal theories directly (without normalisation to clausal form). Fortunately, the extension of Satchmo, Satchmo-1 [2], can deal with such formulas directly. A second problem is that Satchmo and Satchmo-1 were not designed to cope with equality atoms occurring in the head: the generated models satisfy *FEQ* only when no equality atoms occur in the head of the rules. The solution is to treat equality as any other predicate and to add *FEQ* explicitly to the theory. But then a third problem arises: *FEQ* is not in a range-restricted form. Satchmo-1 can only handle range-restricted formulas. However, any theory can be transformed to the range-restricted form. After performing this transformation, and without dealing with the technical details of the computation, one may obtain the computation tree presented in Fig. 5.

Globally, the structure of the SLD + abduction tree of Fig. 4 can still be seen in the Satchmo-1 tree. Striking is the *duality* of variables in the abductive derivation and skolem constants in the model generation. However, one difference is that the Satchmo-1 tree comprises many additional inference steps due to the application of the axioms of *FEQ*. In the abductive derivation, the additional steps correspond to the unification operation (e.g. on both left-most branches, the failure of the unification of $\{X=a, X=b\}$ corresponds to the derivation of the inconsistency of the facts $\{sk_1=a, sk_1=b\}$).

Another difference is that the generated model

$$\{p(a,a), q(a,sk_2), p(sk_1,a), p(a,sk_1), p(sk_1,sk_1), q(sk_1,sk_2), sk_1=a, a=sk_1,$$

$$a=a, sk_1=sk_1, sk_2=sk_2\}$$

is much larger than the model which is dual to the abductive solution. Satchmo-1 generates, besides the atoms of this model, also all logical implications of *FEQ*, comprising all substitutions of a by sk_1. It is clear that in general this will lead to an exponential explosion.

However, observe that we obtain the desired model by *contracting* sk_1 and a in the generated model. Therefore, extending Satchmo-1 with methods for dynamic contraction of equal elements would solve the efficiency problem and would restore the duality on the level of the declarative semantics.

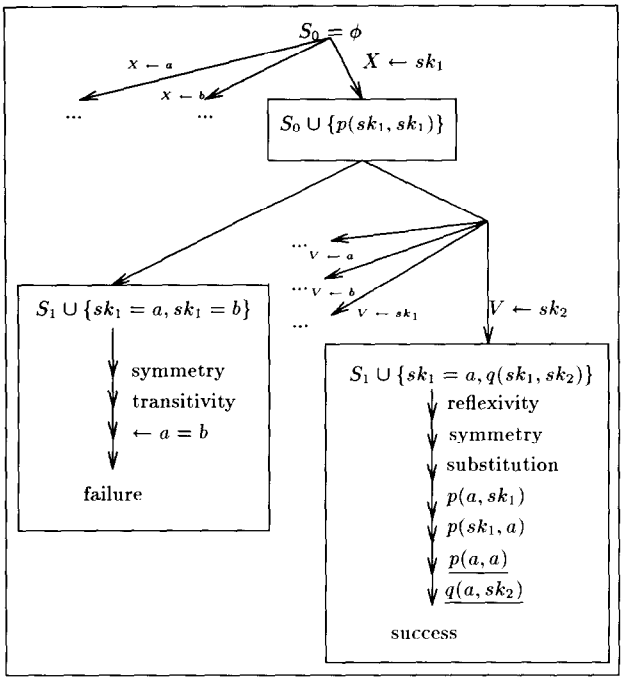

Fig. 5. Execution tree of Satchmo-1.

Contraction of a model is done by taking one unique witness out of every equivalence class of equal terms and replacing all terms in the facts of the model by their witnesses. In turns out that techniques studied in term rewriting are useful to implement this. The idea is to consider the set of inferred equality facts as a term rewriting system (TRS), to transform the set to an equivalent *complete* TRS which then allows to *normalise* all terms in the model. Normalisation is a procedural way to replace terms by their witnesses.

However, a problem with the completion and normalisation procedures in term rewriting is that they are developed for standard equality (*EQ*) instead of *FEQ*. This has led us to develop a framework for model generation with a generic underlying equality theory (Section 4). The framework is based on generalised notions of *completion* and *normalisation* w.r.t. an arbitrary equality theory. Two instances of the framework have been implemented (Section 6). One instance is a model generator for *EQ*, obtained by embedding existing completion and normalisation techniques from term rewriting in Satchmo-1. The second instance is a model generator for *FEQ*. It is based on the completion and normalisation procedures which we developed for *FEQ* (Section 5). At first sight, these procedures may seem alien to logic programming, but the contrary is true: they restore the broken duality between SLD+abduction and Satchmo-1:

- the completion procedure corresponds dually to unification. The dual of the mgu (by replacing variables by skolem constants) is the completion of the set of equality atoms.
- the normalisation corresponds dually to applying the mgu.

Therefore, incorporating these techniques in Satchmo-1 also restores the duality on the level of procedural semantics.

The starting goal for the research reported here was to investigate the duality between abduction and model generation. This goal led us to a second goal, valuable in its own right, namely the extension of current techniques for model generation with efficient treatment of equality. The paper presents answers to both goals. There are other spin-offs. An illustration of this is found in the context of planning as abduction in the event calculus. The event calculus contains a clause, expressing that a property holds at a certain moment if there is an earlier event which initiates this property, and the property is not terminated (clipped) in between:

$$holds\_at(P, T) \leftarrow happens(E), initiates(E, P),$$
$$E < T, \neg\, clipped(E, P, T).$$

A planner uses this clause to introduce new events which initialise some desired property. Technically, this is done by first skolemising and then abducing the *happens* goal. However, skolemisation requires explicit treatment of the equality predicate as an abducible satisfying *FEQ* [11]. The techniques proposed in this paper allow an efficient treatment of the abduced equality atoms and provide a declarative semantics for it.

The paper is structured as follows. In Section 2, we present the class of theories for which the model generation is designed. Section 3 recalls and extends the basic concepts of term rewriting. In Section 4, the framework for model generation is presented and important semantic results are formulated. In Section 5, the duality with abductive reasoning is formalised. Section 6 is about the implementation of the framework. Section 7 discusses future and related work.

2. Extended programs

In this section we introduce the formalism for which the model generation will be designed. This formalism should at least contain any theory that can be obtained as the only-if part of the definition in the Clark-completion of definite logic programs. The extended clause formalism introduced below generalises, both, this kind of formulas and those in clausal form.

Definition 2.1. Let $\mathscr{L}$ be a first-order language. An *extended clause* or *rule* is a closed formula of the type:

$$\forall(G_1, \ldots, G_k \rightarrow E_1, \ldots, E_l),$$

where E_i has the general form

$$\exists Y_1, \ldots, Y_m\colon\ s_1 = t_1 \wedge \cdots \wedge s_g = t_g \wedge F_1 \wedge \cdots \wedge F_h$$

such that all G_i are atoms based on $\mathscr{L}$, all F_i are nonequality atoms based on $\mathscr{L}$.

Definition 2.2. An extended program is a set of extended clauses.

Interestingly, the extended clause formalism can be proved to provide the full expressivity of first-order logic. Any first-order logic theory can be translated to a logically equivalent extended program, in the sense that they share exactly the same models. (Recall that the equivalence between a theory and its clausal form is much weaker: the theory is consistent iff its clausal form is consistent.) We refer to the appendix for the proof of this result.

In the sequel, the standard theory of equality for a first-order language $\mathscr{L}$ will be denoted $EQ(\mathscr{L})$. It is the following theory:

$$\forall X\colon\ X = X,$$

$$\forall X, Y\colon\ X = Y \leftarrow Y = X,$$

$$\forall X, Y, Z\colon\ X = Z \leftarrow X = Y,\ Y = Z.$$

For each functor f/n in $\mathscr{L}(n > 0)$:

$$\forall X_1, \ldots, X_n, Y_1, \ldots, Y_n\colon\ f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n)$$
$$\leftarrow X_1 = Y_1, \ldots, X_n = Y_n.$$

For each nonequality predicate symbol p/n in $\mathscr{L}$:

$$\forall X_1, \ldots, X_n, Y_1, \ldots, Y_n\colon\ p(X_1, \ldots, X_n) \leftarrow p(Y_1, \ldots, Y_n),$$
$$X_1 = Y_1, \ldots, X_n = Y_n.$$

Note that if $\mathscr{L}'$ is an extension of $\mathscr{L}$ by constants then $EQ(\mathscr{L}')$ is identical to $EQ(\mathscr{L})$. The theory of free equality or Clark equality will be denoted $FEQ(\mathscr{L})$ [3]. It extends $EQ(\mathscr{L})$ by the following constraints:

For each functor f/n in $\mathscr{L}(0 < n)$ and for each i $(1 \leqslant i \leqslant n)$:

$$\forall X_1, \ldots, X_n, Y_1, \ldots, Y_n\colon\ f(X_1, \ldots, X_n) = f(Y_1, \ldots, Y_n) \rightarrow X_i = Y_i.$$

For each pair of different functors g/m and f/n in $\mathscr{L}$ $(n, m \geqslant 0)$:

$$\forall X_1, \ldots, X_m, Y_1, \ldots, Y_n\colon\ \neg\, g(X_1, \ldots, X_m) = f(Y_1, \ldots, Y_n).$$

For each t, a term which contains the variable X:

$$\forall(\neg\, t = X).$$

Observe that both $EQ(\mathscr{L})$ and $FEQ(\mathscr{L})$ are Horn clause theories. A theory T, based on $\mathscr{L}$, is called a *theory with equality* if it comprises $EQ(\mathscr{L})$. A theory T, based on $\mathscr{L}$ is called an *equality theory* if it is a theory with equality in which $=$ is the only predicate symbol in all formulas, the substitution axioms of $EQ(\mathscr{L})$ excepted.

3. Concepts of term rewriting

The techniques we intend to develop for dealing with equality are inspired by term rewriting. However, work in this area is too restricted for our purposes, because the concepts and techniques assume the general equality theory EQ underlying the term rewriting. To be able to deal with FEQ, we extend the basic concepts for the case of an arbitrary underlying equality theory E. In the sequel, equality and identity will be denoted distinctly when ambiguity may occur, respectively, by $=$ and $\equiv$. For an overview of the basic notions of TR, see [10]. We recall the general ideas.

Definition 3.1. Let $\mathscr{L}$ be a first-order language. A reduction rule based on $\mathscr{L}$ is of the form $s \rightarrow t$, where

- s, t are terms based on $\mathscr{L}$,
- s is not a variable,
- all variables in t are contained in s.

The logical meaning of a reduction rule $s \rightarrow t$ is: "$\forall(s=t)$". Procedurally, these axioms are used in one direction: a term containing the left-hand term of a reduction rule is *reduced* by replacing this subterm by the right-hand term.

Definition 3.2. A term rewriting system (TRS) based on $\mathscr{L}$ is a finite set of reduction rules based on $\mathscr{L}$.

In what follows, term rewriting systems are denoted by the symbols γ and δ. We denote the Herbrand universe and Herbrand base based on a language $\mathscr{L}$ as $HU(\mathscr{L}), HB(\mathscr{L})$, respectively, and the least Herbrand model of a theory T based on $\mathscr{L}$ as $LHM(\langle \mathscr{L}, T \rangle)$.

With a term rewriting system, a reduction relation between terms can be associated.

Definition 3.3. The reduction relation $\xrightarrow{\gamma, \mathscr{L}}$ is a binary relation on the set of all terms based on $\mathscr{L}$ such that $t_1 \xrightarrow{\gamma, \mathscr{L}} t_2$ iff there exists a reduction rule $s \rightarrow t$ and a substitution θ such that $s.\theta$ occurs in t_1 and t_2 is obtained by replacing $s.\theta$ by $t.\theta$.

Definition 3.4. The reflexive, symmetric, transitive closure of $\xrightarrow{\gamma, \mathscr{L}}$ is denoted $\xleftrightarrow{\gamma, \mathscr{L}}{}^*$. Its restriction to the set of ground terms $HU(\mathscr{L})$ is denoted $\overset{\gamma, \mathscr{L}}{=\!=}$.

Observe that $EQ(\mathscr{L})+\gamma$ is a definite program and, hence, has a least Herbrand model [26]. This is the set of all ground atoms based on $\mathscr{L}$ which are logical consequences of $EQ(\mathscr{L})+\gamma$. The following property indicates the relationship with $\overset{\gamma,\mathscr{L}}{=}$.

Proposition 3.5. $\overset{\gamma,\mathscr{L}}{=}$ *is* $LHM(\langle \mathscr{L}, EQ(\mathscr{L})+\gamma\rangle)$.

Proof. It is easy to see that $\overset{\gamma,\mathscr{L}}{=}$ satisfies all the axioms of $EQ(\mathscr{L})+\gamma$. Since it is a Herbrand interpretation, $LHM(\langle \mathscr{L}, EQ(\mathscr{L})+\gamma\rangle)\subseteq \overset{\gamma,\mathscr{L}}{=}$. Conversely, the interpretation of $=$ in $LHM(\langle \mathscr{L}, EQ(\mathscr{L})+\gamma\rangle)$ is reflexive, symmetric and transitive and if $s\xrightarrow{\gamma,\mathscr{L}}t$ then $s=t\in LHM(\langle \mathscr{L}, EQ(\mathscr{L})+\gamma\rangle)$. Hence, $\overset{\gamma,\mathscr{L}}{=}\subseteq LHM(\langle \mathscr{L}, EQ(\mathscr{L})+\gamma\rangle)$. □

A term rewriting system associates with every term a reduction tree: a tree of terms such that t_2 is a son of t_1 iff $t_1\xrightarrow{\gamma,\mathscr{L}}t_2$. Such a tree corresponds to all possible reductions of the root term. A *noetherian* term rewriting system has the property that each reduction tree is finite. A *confluent* or *Church–Rosser* term rewriting system has the property that, for any pair of nodes in any reduction tree, there exist paths leaving from this nodes, and leading to an identical term. From the procedural point of view, a *noetherian* and *Church–Rosser* TRS satisfies the desirable property that the reduction tree of any term t is finite and that all its leaves are labelled by the same term, called the normal form of t and denoted $t.\gamma$. This *normalisation* operation can be extended to atoms, formulas and sets of these in a natural way. A term which cannot be reduced any further is called *normal.* In term rewriting, a noetherian and Church–Rosser TRS is called *complete.* Below, we extend this concept for an underlying equality theory E.

Definition 3.6. Let E be an equality theory based on a language $\mathscr{L}$, and γ a term rewriting system based on $\mathscr{L}$. γ is complete w.r.t. E iff γ is noetherian and Church–Rosser and, moreover, for each language $\mathscr{L}'$ extending $\mathscr{L}$ with constants, $\langle \mathscr{L}', E+\gamma\rangle$ has a least Herbrand model, which is the set of ground atoms $s=t$ constructed from terms in $HU(\mathscr{L}')$ such that $s.\gamma\equiv t.\gamma$.

The introduction of the extension $\mathscr{L}'$ of $\mathscr{L}$ in the third condition assures that the property of being complete is language-independent. This will prove to be important for the remainder of the paper, because during model generation skolem constants are introduced dynamically.

This definition extends the normal definition in term rewriting by the third condition. However, for $E=EQ$, it was proved in [12] that this condition is implied by the noetherian and Church–Rosser properties (see also Proposition 3.7(a)). This is not the case for an arbitrary equality theory (as FEQ). For example, take any language

comprising constants a and b and take $E = FEQ(\mathscr{L})$. Define $\gamma = \{a \rightarrow b\}$. γ is noetherian and Church–Rosser but $FEQ + \gamma$ is inconsistent and hence has no least Herbrand model.

Much work in term rewriting concentrates on complete TRSs (with of course EQ as underlying equality theory). One of the central themes in TR is the *validity problem*: given some TRS γ and terms s, t, decide whether $EQ(\mathscr{L}) + \gamma \models \forall(s = t)$. In general, the validity problem is undecidable but for a complete TRS γ, it is decidable since $EQ(\mathscr{L}) + \gamma \models \forall(s = t)$ if and only if $s.\gamma \equiv t.\gamma$. This interesting result has motivated the research in TR to develop methods to transform a TRS into a logically equivalent (w.r.t. EQ) but complete TRS. This operation is called the *completion*. The best-known and oldest completion algorithm was proposed by Knuth and Bendix [15]. The problem of finding a complete TRS for a given set of equations w.r.t. EQ, is in general unsolvable (otherwise, the validity problem could be solved). However, the completion of a *ground TRS* (w.r.t. EQ) can be computed [9]. That suffices for our purposes, since during model generation only ground instances of rules are applied; hence, the equality sets to be completed are always ground.

Recall from Section 1 that our main goal was to introduce *dynamic contraction* during model generation. The main reason why here we introduce the notion of a complete TRS is not to solve some validity problem but because a complete TRS γ allows to contract the partially constructed model. Indeed, because two terms equal w.r.t. $EQ(\mathscr{L}) + \gamma$ have the same normalisation, and since a term and its normalisation occur obviously in the same equivalence class, it follows directly that each equivalence class contains precisely one normal term. Hence, the normal terms can be taken as the unique witnesses, and normalisation is the procedure to contract models by replacing terms by their unique witnesses.

In the following proposition, some basic properties of complete TRSs are explored.

Proposition 3.7. (a) *If γ is noetherian and Church–Rosser, then γ is complete w.r.t. $EQ(\mathscr{L})$.*

(b) *If γ is complete w.r.t. E then γ is complete w.r.t. $EQ(\mathscr{L})$.*

(c) *If γ is complete w.r.t. E then for each $\mathscr{L}'$ extending $\mathscr{L}$ with constants:*

$$LHM(\langle \mathscr{L}', E + \gamma \rangle) = LHM(\langle \mathscr{L}', EQ(\mathscr{L}) + \gamma \rangle).$$

Proof. For a proof of (a), define $S = \{s = t \mid s, t \in HU(\mathscr{L}') \text{ and } s.\gamma \equiv t.\gamma\}$. We show that S is $\overset{\gamma, \mathscr{L}'}{=\!=}$. First observe that if $s \xrightarrow{\gamma, \mathscr{L}'} t$ then the reduction tree of t occurs in the reduction tree of s, so $s.\gamma \equiv t.\gamma$. Hence S subsumes $\xrightarrow{\gamma, \mathscr{L}'}$. Further on, it is trivial to see that S defines a reflexive, symmetric and transitive relation, hence $\overset{\gamma, \mathscr{L}'}{=\!=} \subseteq S$.

Conversely, clearly each atom $s = (s.\gamma)$ is an element of $\overset{\gamma, \mathscr{L}'}{=\!=}$. Because of this and the symmetry and transitivity of $=$ in $\overset{\gamma, \mathscr{L}'}{=\!=}$, for any pair of terms s, t based on $\mathscr{L}'$, $s.\gamma \equiv t.\gamma$ implies $s = t \in \overset{\gamma, \mathscr{L}'}{=\!=}$. So, $S \subseteq \overset{\gamma, \mathscr{L}'}{=\!=}$.

Since by Proposition 3.5, $\overset{\gamma,\mathscr{L}'}{=}$ is $LHM(\langle \mathscr{L}', EQ(\mathscr{L})+\gamma\rangle)$, we find that $s=t \in LHM(\langle \mathscr{L}', EQ(\mathscr{L})+\gamma\rangle)$ iff $s.\gamma \equiv t.\gamma$. Hence, γ is complete w.r.t. $EQ(\mathscr{L})$. (b) is a direct consequence of (a): a complete TRS γ w.r.t. E is noetherian and Church–Rosser, so by (a) γ is complete w.r.t. $EQ(\mathscr{L})$. (c) follows also directly from (a). If γ is complete w.r.t. E then by definition $LHM(\langle \mathscr{L}', E+\gamma\rangle)=S$. By (a), we have that $S=LHM(\langle \mathscr{L}', EQ(\mathscr{L})+\gamma\rangle)$. □

Definition 3.8. A completion of a TRS γ w.r.t. $\langle \mathscr{L}, E\rangle$ is
- $\{\square\}$ if $\langle \mathscr{L}, E+\gamma\rangle$ is inconsistent,
- a complete TRS γ_c based on $\mathscr{L}$, such that $\langle \mathscr{L}, E\rangle \models \gamma \leftrightarrow \gamma_c$.

We denote the completion of γ by TRS-comp(γ).

Our framework for model generation is developed for logical theories consisting of two components, an extended program P and an *underlying* equality theory E. This distinction reflects the fact that the model generation mechanism applies only to the extended clauses of P, while E is dealt with in a procedural way, using completion and normalisation. E has to satisfy the conditions of the following definition.

Definition 3.9. An *equality theory with completion* E based on a language $\mathscr{L}$, is a clausal equality theory equipped with a completion procedure which, for each ground TRS based on an extension of $\mathscr{L}$ by constants, produces a ground completion.

In Section 5, we will prove that $FEQ(\mathscr{L})$ is an equality theory with completion, and that its completion procedure is dual to the unification procedure. As indicated earlier, $EQ(\mathscr{L})$ has also a completion procedure, which is a variant of the Knuth–Bendix procedure. Hence, $EQ(\mathscr{L})$ is an equality theory with completion.

Another concept taken from term rewriting is *E-unification.*

Definition 3.10. Let t, s be terms, γ a TRS. An E-unifier θ of t and s w.r.t. γ is a substitution such that $EQ+\gamma \models \forall(s.\theta=t.\theta)$. An E-unifier of atoms $p(t_1,\ldots,t_n)$ and $p(s_1,\ldots,s_n)$ is an E-unifier of the tuples $(t_1,s_1),\ldots,(t_n,s_n)$. An E-unifier of a set of pairs of atoms is an E-unifier of all the pairs of atoms.

4. A framework for model generation

Informally, a model generator constructs a sequence[1] $(Cl_d, j_d)_1^n$, where Cl_d is the ground instance of a rule applied at the dth inference step, and j_d the index indicating the conclusion of Cl_d that was selected, an increasing sequence of sets of *asserted*

[1] $(A_d)_i^n$ denotes a sequence $(A_i,\ldots,A_n)$.

ground facts $(M_d)_0^n$ of non-equality predicates, a sequence of complete term rewriting systems $(\gamma_d)_0^n$, each of which is equivalent with the set of *asserted* equality facts, and an increasing sequence of sets of skolem constants $(Sk_d)_0^n$, obtained by skolemising the existentially quantified variables.

Below, a substitution is called normal w.r.t. some TRS γ if it assigns normal terms to each variable. The normalisation of a substitution θ is the substitution obtained by normalising all right-hand terms in θ. We denote the normalised substitution by $\theta.\gamma$. A substitution is called grounding for some open formula if it assigns to each free variable a ground term. An instance of an extended clause is obtained by applying a grounding substitution for the extended clause. A normal instance is an instance obtained by applying a normal substitution. Note that this does not imply that all terms in the normal instance are normal. Only the terms assigned to variables are normal.

Definition 4.1. Let $\mathscr{L}$ be a language, $\mathscr{L}_{sk}$ an infinite countable alphabet of skolem constants, not occurring in $\mathscr{L}$, T a theory based on $\mathscr{L}$ and consisting of an extended program P and of an equality theory E with completion, equipped with completion procedure *TRS*-comp.

A nondeterministic model generator with equality (NMGE) K is a tuple of four sequences $(Sk_d)_0^n, (M_d)_0^n, (\gamma_d)_0^n$ and $(Cl_d, j_d)_1^n$ where $n \in \mathbb{N} \cup \{\infty\}$. The sequences satisfy the following conditions:

(1) $M_0 = Sk_0 = \{\ \}; \gamma_0 = \text{TRS-comp}(\{\ \})$.

(2) For each d such that $0 < d \leqslant n$, Cl_d, j_d, Sk_d, M_d and γ_d are obtained from Sk_{d-1}, M_{d-1} and γ_{d-1} by applying the following steps:

(a) Selection of rule and conclusion: Select nondeterministically a rule C of P and a substitution θ such that the following conditions hold (let C be of the form $G_1, \ldots, G_k \rightarrow E_1, \ldots, E_l$):

- θ is a grounding substitution of C, normal w.r.t. γ_{d-1} and based on $\mathscr{L} + Sk_{d-1}$.
- there exist atoms $A_1, \ldots, A_k$ from M_{d-1} such that θ is an E-unifier of the set $\{(G_1, A_1), \ldots, (G_k, A_k)\}$ w.r.t. γ_{d-1}.

Define Cl_d as $(C.\theta)$. If $l = 0$, define $Sk_d = \{\ \}$, $M_d = \gamma_d = \{\square\}$ and $n = d$. Otherwise, select nondeterministically a conclusion $E_j.\theta$ from the head of Cl_d. Define $j_d = j$. We say that the rule Cl_d applies with its j_dth conclusion.

(b) Skolemisation: Let $E_{j_d}.\theta$ be of the form $\exists Y_1, \ldots, Y_m$: $s_1 = t_1 \wedge \cdots \wedge s_g = t_g \wedge F_1 \wedge \cdots \wedge F_h$. Replace $Y_1, \ldots, Y_m$ by fresh skolem constants $sk_1, \ldots, sk_m$ from $\mathscr{L}_{sk} \backslash Sk_{d-1}$. Define $Sk_d = Sk_{d-1} \cup \{sk_1, \ldots, sk_m\}$.

(c) Completion: Define $\gamma_d = TRS\text{-}comp(\gamma_{d-1} + \{s_1 = t_1, \ldots, s_g = t_g\})$. If γ_d is $\{\square\}$ then define $M_d = \{\square\}$ and $n = d$.

(d) Normalisation + assertion: Define $M_d = M_{d-1}.\gamma_d + \{F_1, \ldots, F_h\}.\gamma_d$, obtained by computing the normal form of all facts in these sets.

K is failed if n is finite and $\gamma_n = M_n = \{\square\}$. This situation occurs when Cl_n is a negative clause, or when $E + \gamma_{n-1} + \{s_1 = t_1, \ldots, s_g = t_g\}$ is inconsistent.

If K is not failed then K is called successful.

Note that (a) requires an E-unifier θ of the body of the rule C and facts of M_{d-1}. In Proposition 5.7, we will show that with FEQ as underlying equality theory with completion, E-unification collapses to unification, i.e. θ is an E-unifier iff θ is a unifier.

Example. Take EQ as underlying equality theory with completion and consider the following theory:

$$\rightarrow a = f(a),$$

$$p(X) \rightarrow a = X,$$

$$\rightarrow p(b).$$

An NMGE is obtained as follows. In the first step, the first rule is selected and the completion of $a=f(a)$ is computed. This returns $\gamma_1 = \{f(a) \rightarrow a\}$. In the second step, $p(b)$ is derived via the third rule. $p(b)$ is in normal form and $M_2 = \{p(b)\}$. In the third step, $a=b$ is derived via the second rule. Now, we must compute the completion of $\{f(a)=a, a=b\}$. A solution is $\gamma_3 = \{f(a) \rightarrow a, b \rightarrow a\}$. With this TRS, M_2 is normalised to $M_3 = \{p(a)\}$. At this point, all rules are satisfied, and the computation terminates with a finite model with domain $D = \{a\}$, with $f/1$ the function which maps a on a, with $p = \{p(a)\}$ and with $=$ the identity relation on D. Note that a model generator without special treatment for equality will loop. During this loop, an infinite number of facts will be derived: for each n and m: $p(f^n(a)), p(f^n(b))$, $f^n(a) = f^m(b)$, etc., are logical implications and will be derived.

Now assume that we add the axiom $p(f(f(f(f(f(a)))))) \rightarrow$. The previous NMGE must be extended by a fourth step. In this fourth step, the E-unifier between $p(f^5(a))$ and $p(a)$ w.r.t. γ_3 is computed. The empty substitution is an E-unifier between these atoms, and we obtain failure. Note that a model generator without special treatment of equality will also eventually stop, but this will last until $p(f^5(a))$ is derived by application of the axioms of EQ. In general, a high number of other useless atoms will be derived before.

Finally, observe that if FEQ was the underlying equality theory, then failure would occur when the rule $\rightarrow a = f(a)$ is selected. This atom is inconsistent with the occur-check axioms.

Example. Take FEQ as underlying equality theory and consider the following theory:

$$\exists X\colon\ p(f(h(X), X)) \wedge q(X),$$

$$p(X) \rightarrow (\exists Z\colon\ X = f(Z, g(a))),$$

$$q(g(Z)) \rightarrow.$$

An NMGE selects first the first clause. The variable X is skolemised and the two atoms are asserted. This produces a first model M_1 of the form $\{p(f(h(sk_1), sk_1)), q(sk_1)\}$. In the second step, the second rule is applied, deriving the equality atom

$f(h(sk_1), sk_1) = f(sk_2, g(a))$. The completion of this atom is obtained by applying a dual form of unification, this yields $\gamma_2 = \{sk_1 \rightarrow g(a), sk_2 \rightarrow h(g(a))\}$. After normalisation, we obtain $M_2 = \{p(f(h(g(a)), g(a))), q(g(a))\}$. In the third step, $q(g(a))$ can be plugged in the third rule, and failure occurs.

One remark to be made here is that since the language comprises the functor $f/1$, FEQ comprises an infinite number of disequality axioms. Hence, it is impossible to use a model generator without special treatment of FEQ. A second remark is that the above theory is consistent under EQ. Indeed, a completion under EQ of the equality fact $f(h(sk_1), sk_1) = f(sk_2, g(a))$ under EQ is $\{f(h(sk_1), sk_1) \rightarrow f(sk_2, g(a))\}$. From this TRS, $sk_1 = g(a)$ cannot be derived. Therefore, the third rule cannot be applied.

Below, LHM_d denotes the least Herbrand model of $\langle \mathscr{L} + Sk_d, EQ(\mathscr{L}) + M_d + \gamma_d \rangle$.

Proposition 4.2. *$(LHM_d)_0^n$ is a monotonically increasing sequence.*

The proof of this proposition uses the next lemma.

Lemma 4.3. *Let E be an equality theory with completion, based on $\mathscr{L}$. Let γ be a term rewriting system, $\delta = TRS\text{-}comp(\gamma)$, M a set of ground nonequality atoms, normal w.r.t. δ, and γ, δ and M based on $\mathscr{L} + Sk$.*

For each extension $\mathscr{L}'$ of $\mathscr{L} + Sk$ by constants, the following equalities hold:

(a) $LHM(\langle \mathscr{L}', E + \gamma + M \rangle) = LHM(\langle \mathscr{L}', E + \delta + M \rangle)$,

(b) $LHM(\langle \mathscr{L}', E + \delta + M \rangle) = LHM(\langle \mathscr{L}', EQ(\mathscr{L}) + \delta + M \rangle)$,

(c) $LHM(\langle \mathscr{L}', EQ(\mathscr{L}) + \delta + M \rangle) = \overset{\delta, \mathscr{L}'}{=\!=} \cup \{A \mid A \in HB(\mathscr{L}') \text{ and } A.\delta \in M\}$.

Proof. The lemma follows straightforwardly from the equivalence of γ and δ w.r.t. E, Propositions 3.5 and 3.7 and the substitution axioms for the predicates. □

Proof of Proposition 4.2. We have that

$$
\begin{aligned}
LHM_d &= LHM(\langle \mathscr{L} + Sk_d, EQ(\mathscr{L}) + M_d + \gamma_d \rangle) \\
&\subseteq LHM(\langle \mathscr{L} + Sk_{d+1}, EQ(\mathscr{L}) + M_d + \gamma_d \rangle) \\
&= LHM(\langle \mathscr{L} + Sk_{d+1}, E + M_d + \gamma_d \rangle) \quad \text{(Lemma 4.3(b))} \\
&\subseteq LHM(\langle \mathscr{L} + Sk_{d+1}, E + M_d + \{F_1, \ldots, F_h\} + \gamma_d \\
&\qquad + \{s_1 = t_1, \ldots, s_g = t_g\} \rangle) \\
&= LHM(\langle \mathscr{L} + Sk_{d+1}, E + M_{d+1} + \gamma_{d+1} \rangle) \quad \text{(Lemma 4.3(a))} \\
&= LHM_{d+1} \quad \text{(Lemma 4.3(b))}. \qquad \square
\end{aligned}
$$

An NMGE performs a fixpoint computation, the result of which can be seen as an interpretation of the language $\mathscr{L}$ and, as we later show, as a model of $\langle \mathscr{L}, P+E \rangle$.

Definition 4.4. The skolem set used by an NMGE K is $\bigcup_0^n Sk_d$ and is denoted by $Sk(K)$. $\bigcup_0^n LHM_d$ defines a Herbrand interpretation for the language $\mathscr{L}+Sk(K)$ and is denoted $K\uparrow$. The non-Herbrand interpretation of $\mathscr{L}$ obtained by restricting $K\uparrow$ to the symbols of $\mathscr{L}$ is denoted by $K\uparrow_{\mathscr{L}}$ and is defined as follows:

- domain: $HU(\mathscr{L}+Sk(K))$,
- for each functor f/n of $\mathscr{L}$ $(n \geqslant 0)$: $K\uparrow_{\mathscr{L}}(f/n)$ is the function which maps terms $t_1, \ldots, t_n$ of $HU(\mathscr{L}+Sk(K))$ to $f(t_1, \ldots, t_n)$.
- for each predicate of $\mathscr{L}$: $K\uparrow_{\mathscr{L}}(p/n)$ is the set of $p(t_1, \ldots, t_n)$ facts in $K\uparrow$.

Corollary 4.5. *If K is a finite successful NMGE of length n, then $K\uparrow = LHM_n$.*

Not all NMGEs generate models of $P+E$. For example, the empty NMGE (({ }),({ }), (TRS-comp({ })),( )) trivially satisfies the definition of an NMGE, but will not generate a model if P contains one positive extended clause, i.e. an extended clause with empty body. In that case the empty NMGE is an example of an unfair NMGE: there exists a rule with a true body, but which is never applied.

Definition 4.6. A NMGE K is fair iff K is failed or else if the following condition is satisfied: If $Cl = G_1, \ldots, G_k \rightarrow E_1, \ldots, E_l$ is a ground instance of a rule of P, and there exists a d such that Cl is based on $\mathscr{L}+Sk_d$ and the body of Cl holds in LHM_d then there exists a d' such that $E_1 \vee \cdots \vee E_l$ holds in $LHM_{d'}$.

From a procedural point of view, it is uninteresting to apply a rule whose head is satisfied in LHM_{d-1}.

Definition 4.7. An NMGE is redundant iff at least one rule is applied (say at step d) which is satisfied in LHM_{d-1}.

The following proposition establishes a number of basic results which will be used frequently in this and the following section. The first result assures us that if the head of some instance of a rule holds in LHM_d, then it also holds both in each later $LHM_{d'}$ $(d' > d)$ and in $K\uparrow$. Note that the head of an extended clause is a disjunction of existentially quantified conjunctions of atoms, which constitutes the main difference with Proposition 4.2. A second result relates the truth of an instance of an open formula to the truth of the normal instance of the same formula. A third result indicates the role of the E-unification in NMGE: the body of an instance of a rule is true w.r.t. LHM_d iff the instance is obtained by applying an E-unifier of the body of the rule with facts of M_d.

Proposition 4.8. (a) *Let I_1, I_2 be Herbrand interpretations of $\mathscr{L}$ and of an extension $\mathscr{L}'$ of $\mathscr{L}$, respectively, such that $I_1 \subseteq I_2$. Let F be a closed formula based on $\mathscr{L}$ without negation and universal quantifiers. If $I_1 \models F$ then $I_2 \models F$.*

(b) *Let F be an open formula and θ a grounding substitution of F. $LHM_d \models F.\theta$ if and only if $LHM_d \models F.(\theta.\gamma_d)$. Here $\theta.\gamma_d$ denotes the normalisation of θ w.r.t. γ_d.*

(c) *Let $C.\theta$ be a ground instance of a rule based on $\mathscr{L} + Sk_d$. θ is an E-unifier of the body of C and atoms in M_d w.r.t. γ_d iff the body of $C.\theta$ holds in LHM_d.*

Proof. (a) and (b) can easily be proved by induction on the structure of F. Intuitively, what goes wrong with negation and universal quantifiers in (a) is that a new fact in $I_2 \backslash I_1$ may contradict a negative fact in I_1 and a new domain element in $HU(\mathscr{L}') \backslash HU(\mathscr{L})$ may delete a universal property of I_1.

(c) clarifies the role of E-unification appearing in the definition of NMGE. Assume that the body of C is of the form $G_1, \ldots, G_k$. If θ is an E-unifier of G_j and some B in M_d w.r.t. γ_d, for each pair of arguments (t_i, s_i) of $G_j.\theta$ and B, $EQ(\mathscr{L}) + \gamma \models t_i = s_i$. By the substitution axioms, $G_j.\theta$ belongs to LHM_d.

Conversely, assume $G_j.\theta$ belongs to LHM_d. By Lemma 4.3 (c), there exists a B in M_d such that $(G_j.\theta).\gamma_d = B$. Hence, for each pair of terms (t_i, s_i) of $G_j.\theta$ and B, $t_i.\gamma \equiv s_i$. From this it follows that θ is an E-unifier of G_j and B, w.r.t. γ_d. □

Theorem 4.9. (Soundness). *If K is a fair successful NMGE, then $K\uparrow_{\mathscr{L}}$ is a model for $\langle \mathscr{L}, P + E \rangle$ and $P + E$ is consistent (a fortiori).*

We say that $K\uparrow_{\mathscr{L}}$ is the model generated by K.

The proof of the theorem and other proofs below use the following observation.

Lemma 4.10. *Let $\mathscr{L}'$ be an extension of $\mathscr{L}$, T a theory based on $\mathscr{L}$, M' an interpretation of $\mathscr{L}'$, and M the restriction of M' to the symbols of $\mathscr{L}$.*

Then M' is a model of $\langle \mathscr{L}', T \rangle$ iff M is a model of $\langle \mathscr{L}, T \rangle$.

The proof of this lemma is trivial, since the validity of a formula w.r.t. an interpretation depends only on the interpretation of the symbols in the formula.

Proof of Theorem 4.9. First we prove that $K\uparrow_{\mathscr{L}}$ is a model of $\langle \mathscr{L}, E \rangle$. Because of Lemma 4.10, it suffices to show that $K\uparrow$ is a model of $\langle \mathscr{L} + Sk(K), E \rangle$. Consider the two sequences:

$$(A_d)_0^n = (LHM_d)_0^n = (LHM(\langle \mathscr{L} + Sk_d, EQ(\mathscr{L}) + M_d + \gamma_d \rangle))_0^n,$$

$$(B_d)_0^n = (LHM(\langle \mathscr{L} + Sk(K), EQ(\mathscr{L}) + M_d + \gamma_d \rangle))_0^n.$$

Both sequences consist of subsets of the Herbrand base $HB(\mathscr{L} + Sk(K))$. We show that they have the same union, namely $K\uparrow$. One direction follows easily from the fact that, for each d, $A_d \subseteq B_d$; therefore, $K\uparrow = \bigcup_{d=0}^{n} A_d \subseteq \bigcup_{d=0}^{n} B_d$.

For the other direction, we must show that, for each d,

$$LHM(\langle \mathscr{L} + Sk(K), EQ(\mathscr{L}) + M_d + \gamma_d \rangle) \subseteq K\uparrow.$$

Let A be an atom of $LHM(\langle \mathscr{L} + Sk(K), EQ(\mathscr{L}) + M_d + \gamma_d \rangle)$. Because A contains only a finite number of skolem constants, there must be a $d' \geq d$ such that $A \in HU(\mathscr{L} + Sk_{d'})$. Because $E + M_{d'} + \gamma_{d'} \models M_d + \gamma_d$ and by Lemma 4.3(b), it holds that $EQ(\mathscr{L}) + M_{d'} + \gamma_{d'} \models M_d + \gamma_d$. As a consequence, A occurs in $LHM_{d'}$ and hence in $K\uparrow$.

$(B_d)_0^n$ is a monotonically increasing sequence of Herbrand models of $\langle \mathscr{L} + Sk(K), E \rangle$ (Lemma 4.3(b)). A well-known property of clausal theories is that the fixpoint of a monotonically increasing sequence of models is a model. Since E is a clausal theory (by definition of *equality theory with completion*), $K\uparrow$ is a model of $\langle \mathscr{L} + Sk(K), E \rangle$.

It remains to be proved that $K\uparrow$ is a model of P. Assume that there exists a ground instance $G_1, \ldots, G_k \rightarrow E_1, \ldots, E_l$ of a rule of P which is not satisfied by $K\uparrow$. So none of $E_1, \ldots, E_l$ holds in $K\uparrow$, and $G_1, \ldots, G_k$ hold in $K\uparrow$. However, since $(LHM_d)_0^n$ is monotonic, there exists a d such that $G_1, \ldots, G_k$ is in LHM_d. Since K is fair, there is a d' such that at least one E_j holds in $LHM_{d'}$. By Proposition 4.8(a), E_j also holds $K\uparrow$. This contradicts our assumption. □

To state the completeness result, we require an additional concept: the NMGE tree. Analogously with the concept of SLD tree, an NMGE tree is a tree of NMGEs obtained by applying all different conclusions of one rule in the descendants of a node.

Definition 4.11. Let $\mathscr{L}$ be a language, E an equality theory with completion and P an extended program based on $\mathscr{L}$. An NMGE tree (*NMGET*) W for $\langle \mathscr{L}, P + E \rangle$ is a tree such that

- Each node is labelled with a tuple (Sk, M, γ) where Sk is a skolem set, M a set of nonequality facts based on $\mathscr{L} + Sk$, and γ is a ground TRS based on $\mathscr{L} + Sk$.
- To each nonleaf N, a ground instance Cl of a rule of P is associated. For each conclusion with index j in the head of Cl, there is an arc leaving from N which is labelled by (Cl, j). If Cl has no conclusion then one arc leaves with label Cl.
- The sequence of labels on the nodes and arcs on each branch of W constitute an NMGE.

Definition 4.12. An NMGET is fair if each branch is fair.

Definition 4.13. An NMGET is failed if each branch is failed.

Observe that a failed NMGET contains only a finite number of nodes. Also if T is inconsistent then because of the soundness Theorem 4.9, each fair NMGET is failed.

As a completeness result, we want to state that, for any model of $P+E$, the NMGE contains a branch generating a *smaller* model. In a context of Herbrand models, the smaller-than relation can be expressed by set inclusion. However, because of the existential quantifiers and the resulting skolem constants, we cannot restrict to Herbrand models only. In order to define a smaller-than relation for general models, we must have a mechanism to compare models with a different domain. A solution to this problem is provided by the concept of *homomorphism.*

Definition 4.14. Let I_1, I_2 be interpretations of a language $\mathscr{L}$ with domains D_1, D_2.

A homomorphism from I_1 to I_2 is a mapping $h: D_1 \to D_2$ which satisfies the following conditions:

- for each functor f/n $(n \geqslant 0)$ of $\mathscr{L}$ and $x, x_1, \ldots, x_n \in D_1$,

$$x \equiv I_1(f/n)(x_1, \ldots, x_n) \Rightarrow h(x) \equiv I_2(f/n)(h(x_1), \ldots, h(x_n));$$

- for each predicate symbol p/n $(n \geqslant 0)$ of $\mathscr{L}$ and $x_1, \ldots, x_n \in D_1$,

$$I_1(p/n)(x_1, \ldots, x_n) \Rightarrow I_2(p/n)(h(x_1), \ldots, h(x_n)).$$

Intuitively, a homomorphism is a mapping from one domain to another, such that all positive information in the first model is maintained under the mapping. Therefore, the homomorphisms in the class of models of a theory can be used to represent a "*...contains less positive information than...*" relation. We denote the fact that there exists a homomorphism from interpretation I_1 to I_2 by $I_1 \preceq I_2$. This notation captures the intuition that I_1 contains less positive information than I_2.

For NMGETs we can prove the following powerful completeness result.

Theorem 4.15 (Completeness). *Let E be an equality theory with completion and P an extended program, both based on $\mathscr{L}$. Then*

(1) *there exists a fair, nonredundant NMGET for $\langle \mathscr{L}, P+E \rangle$;*

(2) *For each model M of $\langle \mathscr{L}, P+E \rangle$ and each fair NMGET W, there exists a successful branch K of W such that $K \uparrow_{\mathscr{L}} \preceq M$.*

The first item in the proof is concerned with the fairness condition. The condition of fairness is quite strong and is stated in a nonconstructive way. The proof provides a construction of a fair NMGE. This construction is based on the following two lemmas. The first lemma states that after applying a rule, the rule holds. The second lemma constructs a sequence which contains each ground instance of each rule an infinite number of times. This sequence will be used to construct a fair NMGE. Starting from the first clause, for each element in the sequence it is tested on whether or not the rule is violated (not satisfied). A violated rule is applied. In this way, we obtain a fair NMGE, since each rule is tested an infinite number of times and is applied when violated.

Lemma 4.16. *If a ground instance $C.\sigma$ of a rule C is applied at step d, then the conclusion of $C.\sigma$ holds in LHM_d.*

Proof. The straightforward proof is omitted. □

Lemma 4.17. *Let P be an extended program based on $\mathscr{L}$, $\mathscr{L}_{sk}$ a countable alphabet of skolem constants. There exists a countable sequence (C_g) which contains every ground instance of a rule based on $\mathscr{L}+\mathscr{L}_{sk}$ an infinite number of times.*

Proof. A well-known result is that, for any countable set, the set of finite sequences of this set is countable. Each ground instance of a rule is a finite sequence of the countable set of functor symbols of $\mathscr{L}$ and $\mathscr{L}_{sk}$, logical connectors, logical quantifiers, brackets and the period ,. Therefore, the set of possible ground instances of rules is countable.

So there is a sequence (C'_g) which contains each ground instance of a rule of P, based on $\mathscr{L}+\mathscr{L}_{sk}$ at least one time. This sequence can easily be transformed to the desired sequence

$$(C_g)_0^\infty = (C'_1, C'_1, C'_2, C'_1, C'_2, C'_3, C'_1, \ldots)$$

or, more formally, $(C_g)_0^\infty$ is obtained by concatenating the finite sequences $(C'_1, \ldots, C'_n)$ for increasing n. □

Proof of Theorem 4.15. First, using the sequence (C_g) from Lemma 4.17, we construct a fair NMGET. With each node N, starting with the root, an index $g(N)$ is associated which points to the rule in (C_g) whose normalisation is applied to obtain N. The descendants of N are obtained by searching the first rule C_h in (C_g), such that C_h is violated and $h > g(N)$. It is the normalisation of this rule which is applied to obtain the descendants of N and, for each descendant N' of N, $g(N')$ is defined as h. Technically, the construction proceeds by induction on the depth of the nodes:

- The root N_0 is defined as in the definition of NMGET. We define $g(N_0) = -1$.
- Let N_d be a nonfailed node on depth d with index $g(N_d)$. N_d is a leaf of a branch $(N_0, \ldots, N_d)$, with associated sequences $(Sk_i)_0^d, (M_i)_0^d, (\gamma_i)_0^d, (LHM_i)_0^d$ and $(Cl_i, j_i)_1^d$.

Now we look for the first rule C_h in (C_g), such that $h > g(N_d)$, C_h is based on $\mathscr{L}+Sk_d$ and C_h is violated in LHM_d. If such a C_h does not exist anymore, then we obtain a finite fair branch and N_d is a leaf in the constructed tree. If C_h is found, then it is of the form $C.\theta$ where C is a rule of P. Let θ' be $\theta.\gamma_d$. Since $C.\theta$ has a true body, $C.\theta'$ also has a true body w.r.t. LHM_d (Proposition 4.8(b)). By Proposition 4.8(c), θ' is a normal E-unifier w.r.t. γ_d of the atoms in the body of C and facts of M_d. So $C.\theta'$ can be selected. The descendants of N_d are obtained by applying $C.\theta'$ with each conclusion. For each descendant N, we define $g(N) = h$.

It is easy to see that this NMGET is nonredundant: only violated rules are applied. The NMGET is fair: if some rule Cl based on the language of some node N is violated

in LHM_N, this rule reappears in the sequence $(C_g)_{g(N)+1}^{\infty}$ at least one time, say as the hth element ($h > g(N)$). Because g strictly increases for descendants, in each nonfailing branch departing from N, the integrity of C_h will be restored after at most $h-g(N)$ steps: either "by accident" by applying other rules of $C_{g(N)+1}, \ldots, C_{h-1}$ or by applying the normalisation of C_h (Lemma 4.16, Proposition 4.8(b)).

Now we prove the second part of the completeness theorem. The idea of the proof is as follows. We will construct by induction a path K through the NMGET W, such that, for each node N_d on the path, LHM_{N_d} can be mapped into M. At the $(d+1)$th step, the selected rule has a true body in LHM_{N_d} and hence in M. Therefore, one of the conclusions of the selected rule must hold in M. We extend the path by selecting the descendant of N_d corresponding to this conclusion. As a consequence $LHM_{N_{d+1}}$ can again be mapped into M. The resulting branch K returns a fair NMGE such that $K \uparrow_{\mathscr{L}} \preceq M$.

Using W and M, we construct a branch $K = (N_d)_0^n$ in W with corresponding NMGE $(Sk_d)_0^n, (M_d)_0^n, (\gamma_d)_0^n$ and $(Cl_d, j_d)_1^n$ and a sequence of interpretations $(I_d)_0^n$ of $\mathscr{L} + Sk_d$, such that for each d the following invariant relation holds:

(a) I_d extends M by assigning to each skolem in Sk_d an element of the domain of M.

(b) If $d > 0$, then I_d extends I_{d-1} by assigning to each skolem in $Sk_d \backslash Sk_{d-1}$ an element of the domain of M.

(c) I_d is a model of $\langle \mathscr{L} + Sk_d, E + P \rangle$.

(d) I_d is a model of $\langle \mathscr{L} + Sk_d, E + M_d + \gamma_d \rangle$.

The branch is found by induction on the depth d:

- N_0 is the root of W. I_0 is defined as M. Clearly, the invariant relation holds.
- Assume that we have found a path in W of length $d-1$, and N_{d-1} is not a leaf of W. By definition of NMGET, the descendants of N_{d-1} are obtained by applying all the conclusions of the same ground instance Cl_d of a rule C of P:

$$Cl_d = G_1, \ldots, G_k \rightarrow E_1, \ldots, E_l.$$

By Proposition 4.8(c) $G_1, \ldots, G_k$ hold in the least Herbrand model LHM_{d-1} of the theory $\langle \mathscr{L} + Sk_{d-1}, E + M_{d-1} + \gamma_{d-1} \rangle$. By the invariant, I_{d-1} is a model of $\langle \mathscr{L} + Sk_{d-1}, E + M_{d-1} + \gamma_{d-1} \rangle$; therefore, $G_1, \ldots, G_k$ hold w.r.t. I_{d-1}. Since I_{d-1} is a model of P, it is a model of the rule C of which Cl_d is an instance. Therefore, for at least one i, E_i holds w.r.t. I_{d-1}. We select the ith descendant of N_{d-1} as N_d.

Now we extend I_{d-1}. Let E_i be $\exists Y_1, \ldots, Y_m$: $s_1 = t_1 \wedge \cdots \wedge s_g = t_g \wedge F_1 \wedge \cdots \wedge F_h$. Let V be the variable assignment $\{Y_1/a_1, \ldots, Y_m/a_m\}$ such that $s_1 = t_1 \wedge \cdots \wedge s_g = t_g \wedge F_1 \wedge \cdots \wedge F_h$ holds w.r.t. I_{d-1} and V. Let each Y_j be assigned the skolem constant sk_j in N_d. We extend I_{d-1} to I_d by defining $I_d(sk_j) = a_j$.

Clearly, (a) and (b) of the invariant relation hold. (c) is a direct consequence of (a) since I_d is an extension of M (Lemma 4.10). That (d) holds can be seen as follows. I_d is an extension of I_{d-1}; therefore, I_d is a model of $\langle \mathscr{L} + Sk_d, E + M_{d-1} + \gamma_{d-1} \rangle$ (Lemma 4.10). By construction of I_d, I_d is also a model of $\{s_1 = t_1, \ldots, s_g = t_g\}$ and of $\{F_1, \ldots, F_h\}$. One easily verifies that the following proposition holds:

$$E \models (M_{d-1} + \{F_1, \ldots, F_h\} + \gamma_{d-1} + \{s_1 = t_1, \ldots, s_g = t_g\} \Leftrightarrow M_d + \gamma_d).$$

Since I_d is a model of E, I_d is a model of $M_d + \gamma_d$.

The construction above returns a successful branch K in W. Since W is fair, K is a fair NMGE. Because of Theorem 4.9, $K\uparrow_{\mathscr{L}}$ is a model of $\langle \mathscr{L}, P+E\rangle$. It remains to show that there is a homomorphism from $K\uparrow_{\mathscr{L}}$ to M. Because each I_d is consistent with all its predecessors in the sequence, the union of the sequence $(I_d)_0^n$ defines an interpretation I of $\mathscr{L}+Sk(K)$. We use this interpretation to construct a homomorphism h from $K\uparrow_{\mathscr{L}}$ to M. We define h by extending I to all terms in $HU(\mathscr{L}+Sk(K))$. h is defined by induction on the depth of the terms:

- $h(c)=I(c)$ for each constant c in $\mathscr{L}+Sk(K)$.
- $h(f(t_1,\ldots,t_n)=I(f/n)(h(t_1),\ldots,h(t_n))$ for each functor f/n and tuple $t_1,\ldots,t_n$ of terms of $HU(\mathscr{L}+Sk(K))$.

By its construction, h trivially satisfies the first condition of *homomorphism*. The second condition of homomorphism is that, for any atom $p(t_1,\ldots,t_n)\in K\uparrow$, it should hold that $M(p/n)(h(t_1),\ldots,h(t_n))$. Assume $p(t_1,\ldots,t_n)\in K\uparrow$. There exists a d such that $p(t_1,\ldots,t_n)\in LHM_d$. Hence, $E+M_d+\gamma_d\models p(t_1,\ldots,t_n)$ and because of invariant (d), $I_d\models p(t_1,\ldots,t_n)$. Since I is an extension of I_d, $I\models p(t_1,\ldots,t_n)$. This is equivalent with $I(p/n)(h(t_1),\ldots,h(t_n))$ (by definition of truth of an atom w.r.t. to some interpretation). Since I and M interpret p/n by the same relation, it holds that $M(p/n)(h(t_1),\ldots,h(t_n))$. □

The construction of the fair NMGET in Theorem 4.15 does not still give a clue on how to implement the fairness condition in a practical way. This problem will be dealt with properly in Section 6.

As a corollary we obtain the following reformulation of a traditional completeness result.

Corollary 4.18. *If $\langle \mathscr{L}, P+E\rangle$ is consistent then in each fair NMGET there exists a successful branch.*

If there exists a failed NMGET for $\langle \mathscr{L}, P+E\rangle$, then $\langle \mathscr{L}, P+E\rangle$ is inconsistent, and all fair NMGETs are failed.

The completeness result does not imply that all models are generated. For example, for $P=\{p\leftarrow q\}$, the model $\{p,q\}$ is not generated by an NMGE. The following example shows that different NMGETs for the same theory might generate different models.

Example. $P=\{p, q\leftarrow\ \ p\leftarrow\}$. Depending on which of these clauses is applied first, we get two different nonredundant NMGETs. If $p\leftarrow$ is applied first, then $p, q\leftarrow$ holds already and is not applied anymore. So we get an NMGET with one branch of length 1. On the other hand, if $p, q\leftarrow$ was selected first, then two branches exist and we get the solutions $\{p\}$ and $\{p,q\}$.

Therefore, it would be interesting if we could characterise a class of models which are generated by each NMGET. The second item of the completeness Theorem 4.15

gives some indication: for any given model M, some successful branch of the NMGET generates a model with less positive information than M. For the clausal case, models with no redundant positive information are minimal Herbrand models. From this observation one would expect that for a clausal program, each fair NMGET generates all minimal models. Indeed, the following completeness theorem holds.

Theorem 4.19 (Minimal Herbrand models). *If P is clausal, then, for each fair NMGET W, each minimal Herbrand model is generated by a branch in W.*

The proof is easy: for a clausal theory, each successful branch in each fair NMGET W generates a Herbrand model (since no skolemisation is necessary). From Theorem 4.15 it follows that, for each minimal Herbrand model M, there exists a branch K in W such that $K\uparrow \preceq M$. Since for Herbrand models, $\preceq$ corresponds to $\subseteq$, and since M is minimal it follows that $K\uparrow = M$.

Now we return to the general case. Since we have to deal with non-Herbrand models, the concept of *minimal model* must be extended.

Definition 4.20. Let T be a theory based on a language $\mathscr{L}$. A model M_1 *has the same information content* as *or is IC-equivalent with* a model M_2 if there exists a homomorphism from M_1 to M_2 and a homomorphism from M_2 to M_1.

A model M_m of $\langle \mathscr{L}, P\rangle$ is *minimal* iff for each model M' of T such that $M' \preceq M_m$, M' is IC-equivalent with M_m.

The notion of IC-equivalence of models is weaker than the notion of isomorphism. Consider $T = \{p(a) \leftarrow \ \exists X: P(X) \leftarrow\}$. There exist two different nonredundant NMGE-trees, depending on the selection of the first extended clause. Selecting first the rule $p(a) \leftarrow$ one obtains $M_1 = \{p(a)\}$. Selecting the other rule first yields $M_2 = \{p(sk), p(a)\}$. These models are not isomorphic but they are IC-equivalent: $h_1(a) \equiv a$ defines a homomorphism from M_1 to M_2 and $h_2(a) \equiv h_2(sk) \equiv a$ defines a homomorphism from M_2 to M_1.

It is straightforward that *IC-equivalence* defines an equivalence relationship between models and that a model IC-equivalent with a minimal model is minimal also. This definition is a generalisation of the concept of minimality for Herbrand models: for clausal theories, one can easily prove using Theorem 4.19 that a model is minimal iff it is IC-equivalent with a minimal Herbrand model.

Each fair NMGET generates all minimal models, modulo IC-equivalence, but not modulo isomorphism.

Theorem 4.21 (Completeness on minimal models). *Let W be a fair NMGET and C a class of IC-equivalent minimal models. Then there exists a branch of W generating a minimal model of C.*

This theorem is a straightforward consequence of the completeness Theorem 4.15.

In [2], the following completeness theorem for Satchmo-1 was formulated: Satchmo-1 is complete for *finite satisfiability*, i.e. if a theory T has a finite model, then Satchmo-1 generates a finite model. NMGE does not satisfy this property. Consider, for example, the theory $\{\exists X\colon p(a,X)) \leftarrow (\exists Z\colon p(Y,Z)) \leftarrow p(X,Y)\}$. This theory has only an infinite NMGE, generating the model $\{p(a,sk_1), p(sk_1,sk_2), p(sk_2,sk_3), \ldots\}$. However, the interpretation $\{p(a,a)\}$ is a finite model. Satchmo-1 generates both the finite and the infinite model.

This distinction between Satchmo-1 and NMGE is caused by a distinct treatment of existential quantifiers. In an NMGE, each existential variable is skolemised. On the other hand, Satchmo-1 keeps track of the domain of interpretation, and assigns each of the existing domain elements to the existential variable (giving rise to different branches in the computation) before introducing a new skolem constant as a final alternative. Hence, each Satchmo-1 computation tree comprises an NMGET (if no equality atoms in the head occur). As a consequence, the treatment of existential variables as in NMGE is more efficient for showing inconsistency of a theory, whereas the treatment in Satchmo-1 is more suitable for showing consistency of a theory. However, it should be noted that the main issue of this paper, i.e. the technique for dealing with equality, stands orthogonal on the way the existential quantifiers are dealt with. The techniques that are proposed here can as well be incorporated in a procedure which treats existential quantifiers as in Satchmo-1.

5. Duality of SLD + abduction and model generation

The NMGE framework allows to formalise the observations that were made in the introduction. We prove that FEQ is an equality theory with completion and that the completion procedure is dual to the unification procedure. We first introduce the notion of a dualisation more formally.

Definition 5.1. Let $\mathscr{L}$ be a first-order language with variables $\mathscr{L}_V$, $\mathscr{L}_{sk}$ an alphabet of skolem constants which do not occur in $\mathscr{L}$.

A dualisation mapping is a one-to-one correspondence $D\colon \mathscr{L}_{sk} \rightarrow \mathscr{L}_V$.

The dualisation mapping D can be extended to a mapping from $HU(\mathscr{L}+\mathscr{L}_{sk}) \cup HB(\mathscr{L}+\mathscr{L}_{sk})$ to the set of terms based on $\mathscr{L}$ by induction on the depth of terms:

- for each constant c of $\mathscr{L}$: $D(c) \equiv c$,
- for each term or atom $f(t_1, \ldots, t_n)$:

$$D(f(t_1, \ldots, t_n)) \equiv f(D(t_1), \ldots, D(t_n)).$$

D can be further extended to any formula or set of formulas. Under dualisation, a ground TRS γ based on $\mathscr{L}+\mathscr{L}_{sk}$ corresponds to an equation set $D(\gamma)$ with terms based on $\mathscr{L}$.

Definition 5.2. A ground TRS γ is said to be in solved form iff $D(\gamma)$ is an equation set in solved form [20].

An equation set is in solved form iff it consists of equations $X_i = t_i$, such that the X_i's are distinct variables and do not occur on the right-hand side of any equation. So a TRS is in solved form if the left-hand side terms are distinct skolem constants of $\mathscr{L}_{sk}$ which do not occur on the right. A TRS in solved form can also be seen as the dual of an idempotent variable substitution.

Proposition 5.3. *Let γ be a TRS in solved form, then*

(a) *γ is complete w.r.t. $\langle \mathscr{L}, FEQ \rangle$;*

(b) *for each compound term $f(t_1, \ldots, t_n)$, it holds that*

$$f(t_1, \ldots, t_n).\gamma \equiv f(t_1.\gamma, \ldots, t_n.\gamma)$$

and for each constant c of $\mathscr{L}$: $c.\gamma \equiv c$.

Proof. (b) follows straightforwardly from the fact that a TRS in solved form does not contain compound terms on the left.

With respect to (a), it is intuitively clear that, for each term t, its reduction tree is finite and all leaves contain the same term. In other words, γ is noetherian and Church–Rosser. For a formal proof, we can rely on a theorem in term rewriting for *irreducible* TRSs. A TRS γ is called *irreducible* iff, for each $s \to t \in \gamma$, t is in normal form w.r.t. γ and s is in normal form w.r.t. $\gamma \setminus \{s \to t\}$. Clearly, a TRS in solved form is irreducible. It remains to be proved that, for each pair of terms s, t based on an extension $\mathscr{L}'$ of $\mathscr{L}$ by constants, it holds that $s = t \in LHM(\langle \mathscr{L}', FEQ(\mathscr{L}) + \gamma \rangle)$ iff $s.\gamma \equiv t.\gamma$.

Define $S = \{s = t \mid s, t \in HU(\mathscr{L}') \text{ and } s.\gamma \equiv t.\gamma\}$. Since γ is complete w.r.t. $EQ(\mathscr{L})$, S is $LHM(\langle \mathscr{L}', EQ(\mathscr{L}) + \gamma \rangle)$. Since $FEQ(\mathscr{L})$ is an extension of $EQ(\mathscr{L})$, it holds that $S \subseteq LHM(\langle \mathscr{L}', FEQ(\mathscr{L}) + \gamma \rangle)$.

Conversely, one has to prove that S is a Herbrand model of $FEQ(\mathscr{L}) + \gamma$. It suffices to show that S satisfies the axioms in $FEQ(\mathscr{L}) \setminus EQ(\mathscr{L})$:

- Assume $f(t_1, \ldots, t_n).\gamma \equiv f(s_1, \ldots, s_n).\gamma$. By (b) of the proposition, it directly follows that, for each i, $t_i.\gamma \equiv s_i.\gamma$. Hence $t_i = s_i$ belongs to S.
- Because of (b), two terms with distinct main functors f/n and g/m cannot be rewritten to the same term by γ.
- Finally, the consistency of the *occur-check* axioms must be proved. Assume that there exist a pair of terms s, t such that $s.\gamma \equiv t.\gamma$ and s contains t. However, again because of (b), it holds that if s contains t then $s.\gamma$ contains $t.\gamma$. Hence, $s.\gamma$ contains itself. This is impossible. □

Theorem 5.4 expresses the procedural duality between the unification and completion, as announced in Section 1. Here the notion of procedural duality refers to a form

of isomorphism between two procedures. Both the procedures must be decomposable as sequences of basic operations. The isomorphism then refers to the fact that, if the procedures are activated on dual input, then there must be a one-to-one mapping between the two resulting sequences of basic operations, such that the input and output of each two corresponding operators are dual. In the theorem, we take unification as the first procedure with an equality set as input. The dual of the input is the ground TRS obtained by interpreting variables as skolem constants, and the dual of unification is completion.

Theorem 5.4 (Duality of completion and unification). *$FEQ(\mathscr{L})$ is an equality theory with completion. The completion procedure is dual to unification. The dual of the completion of a ground TRS γ based on an extension $\mathscr{L}'$ of $\mathscr{L}$ with constants, is the mgu of $D(\gamma)$. Or $D(TRS\text{-}comp(\gamma))=mgu(D(\gamma))$.*

Proof. Below, the algorithm of [20] is dually reformulated. The symbol x denotes a skolem constant, t a term, E denotes a set of equality atoms. The algorithm proceeds by iteratively transforming a TRS γ by applying the following rewrite rules:

(1) $f(t_1,\ldots,t_n)=f(s_1,\ldots,s_n), E \Rightarrow t_1=s_1,\ldots,t_n=s_n, E$,
(2) $f(t_1,\ldots,t_m)=g(s_1\ldots,s_n), E$, where $f/m \not\equiv g/n \Rightarrow \{\square\}$,
(3) $x=x, E \Rightarrow E$,
(4) $t=x, E$ where t is not a skolem constant $\Rightarrow x=t, E$,
(5) $x=t, E$ where $x \not\equiv t$ and x appears in $t \Rightarrow \{\square\}$,
(6) $x=t, E$ where $x \not\equiv t, x$ does not appear in t, and x has another occurrence in $E \Rightarrow x=t, E.\{x=t\}$.

The algorithm terminates when no rewrite rule can be applied. Its termination follows directly from the termination of the dual algorithm [20]. It returns $\{\square\}$ or a TRS δ in solved form. By Proposition 5.3, it follows that δ is complete. To see that γ and δ are equivalent w.r.t. $FEQ(\mathscr{L})$, just verify that each rewrite rule maintains equivalence w.r.t. $FEQ(\mathscr{L})$. □

We call δ the solved form of γ.

An interesting property of the completion w.r.t. FEQ is that it is *incremental.*

Definition 5.5. The composition operation $\circ$ on term rewriting systems in solved form is defined as the dual of composition of variable substitutions. Or, let γ_1,γ_2 be complete term rewriting systems, dual to the substitutions θ_1,θ_2. Then $\gamma_1\circ\gamma_2$ is defined as $D^{-1}(\theta_1.\theta_2)$.

Proposition 5.6. *Let γ,δ be two ground TRSs, γ_c the completion of γ and δ_c the completion of $\delta.\gamma_c$. Then $\delta_c\circ\gamma_c$ is a completion of $\gamma\cup\delta$.*

Proof. Applying the completion algorithm on $\gamma\cup\delta$, it is always possible first to transform γ into solved form, thus obtaining γ_c. One easily verifies that during this

transformation, δ is transformed gradually to $\delta.\gamma_c$, the normalized form w.r.t. γ_c. So, the result of this first stage is $\gamma_c \cup \delta.\gamma_c$. Then the completion proceeds by bringing $\delta.\gamma_c$ in solved form, which returns δ_c. Similar to the first phase, the effect on the equations of γ_c is that all terms are normalised w.r.t. δ_c. So, the total equation set is $(\gamma_c).\delta_c \cup \delta_c$. This is nothing but $\gamma_c \circ \delta_c$.

Note that this result is dual to the property of unification that if θ is a mgu of an equation set E_1 and σ is the mgu of $E_2.\theta$ (where $E_2.\theta$ denotes the set of equations, obtained by applying θ on both sides of each equation in E_2), then $\theta.\sigma$ is the mgu of $E_1 \cup E_2$. □

Also interesting from a practical point of view is that E-unification w.r.t. a TRS in solved form collapses to unification.

Proposition 5.7. *Let γ be a TRS in solved form and s, t normal terms. θ is a normal E-unifier of (s,t) w.r.t. γ iff θ is a unifier of s and t.*

Proof. Assume θ is a normal E-unifier of (s,t). Hence, $EQ(\mathscr{L}) + \gamma \models \forall(s.\theta = t.\theta)$ and since γ is complete, $s.\theta.\gamma \equiv t.\theta.\gamma$. Since s, t and θ are normal, none of the skolem constants on the left of γ appear in them. Hence $s.\theta \equiv s.\theta.\gamma \equiv t.\theta.\gamma \equiv t.\theta$. So θ is a unifier of s and t.

Conversely, any unifier is a trivial E-unifier. □

A direct consequence of this proposition is that step a in the NMGE process can be simplified by replacing E-unification by unification. Indeed, the terms occurring in M_{d-1} are in normal form w.r.t. γ_{d-1}. Also, all terms in the body of a rule of P are in normal form w.r.t. γ_{d-1} since they are based on $\mathscr{L}$.

As was observed in the introduction, the duality between unification and completion can be extended further to the complete process of SLD + abduction. The latter procedure is a simple extension of SLD resolution for definite abductive programs [6]. Distinction is made between defined predicates which have a definition (i.e. a possibly empty set of definite clauses with head matching the predicate) and abductive predicates which have no definition. An SLD + abduction refutation is a finite SLD derivation during which only atoms of defined predicates are selected for resolution and such that the final resolvent contains only abductive atoms. So, given a definite abductive program P and a definite query Q, we can describe an SLD + abduction derivation of length n, $n \in \mathbb{N} \cup \{\infty\}$, for Q as usual as a triplet of sequences:

- $(R_d)_1^n$ of resolvents with $R_1 = Q$,
- $(C_d)_2^n$ of renamings of program clauses, sharing no variables with each other or with Q,
- $(\theta_d)_2^n$ of substitutions

such that each R_d $(d > 1)$ is derived from R_{d-1} using C_d and θ_d.

Below we assume, without loss of generality, that no $=$ atom occurs neither in a body of a definite clause of P nor in Q. If this special predicate was to occur in P, rename it by a new predicate, for example, by $eq/2$, whose definition consists of the unique clause:

$$eq(X, X) \leftarrow$$

The SLD + abduction procedure takes as input a definite abductive program and definite query. Below we define the dual interpretation of the input. Recall that a query $Q = \leftarrow L_1, \ldots, L_n$ denotes a formula of the form $\forall(\neg L_1 \vee \cdots \vee \neg L_n)$. Therefore, $\neg Q$ denotes the formula $\exists(L_1 \wedge \cdots \wedge L_n)$.

Definition 5.8. Given a definite abductive program P and a definite query Q. We define the dual P_D of (P, Q) as the extended program *only-if* $(P) \cup \{\neg Q\} \backslash A$ where *only-if*(P) consists of the only-if part of every definition in the completion of P and of FEQ, and where $A = \{p(\bar{X}) \rightarrow \mathit{false} \mid p$ is an abducible predicate of $P\}$.

An example of a pair of a program and query and its dual were given in Fig. 2.

Lemma 5.9. *$P_D \backslash FEQ$ is a range-restricted extended program. For each defined predicate p/n of P, one rule C_p occurs in P_D, having $p(\bar{X})$ as the unique atom in its body.*

Range-restricted means that every universal variable which occurs in the head occurs in the body. The lemma is a straightforward consequence of the definition of *only-if*(P).

Below, the duality between SLD + abduction and NMGE suggested by the example in Section 1 is expressed formally. Informally, the selection of a defined atom $p(\bar{t})$ corresponds dually to the selection of the instance of the rule C_p, having the dual of the selected atom in its body. To each clause in the definition of p corresponds a conclusion in C_p. Therefore, we can associate with the selection of a clause the selection of a conclusion of the rule. The unification of the selected atom and the head of the clause corresponds dually to the completion operation of the equality atoms in the selected conclusion. The application of the mgu on resolvent corresponds to the normalisation.

Lemma 5.10. *Let $\mathscr{L}$ be a first-order language, with an alphabet of variables $\mathscr{L}_V$, $\mathscr{L}_{sk}$ an alphabet of skolem constants disjunct from $\mathscr{L}$ and $D: \mathscr{L}_{sk} \rightarrow \mathscr{L}_V$ a dualisation mapping. Let P be a definite abductive program and Q a definite query, both based on $\mathscr{L}$. Let $(R_d)_1^n, (C_d)_2^n$ and $(\theta_d)_2^n$ be an SLD derivation for (P, Q) and $p_d(t_1, \ldots, t_{n_d})$ the atom selected at step d.*

With p_d/n_d a unique rule C_{p_d/n_d} in P_D corresponds, and with the clause C_d, a conclusion E_{j_d} corresponds. Let σ_d be the unifier of the body of C_{p_d/n_d} with $D^{-1}(p_d(t_1, \ldots, t_{n_d}))$.

We define Cl_d, Sk_d, γ_d *and* M_d *as follows:*

$$Cl_1 = \neg Q, \qquad Cl_d = C_{p_d/n_d}.\sigma_d,$$

$$Sk_0 = \{\ \}, \qquad Sk_d = D^{-1}(var(\{Q, C_2, C_3, \ldots, C_i\}),$$

$$\gamma_0 = \gamma_1 = \{\ \}, \quad \gamma_d = D^{-1}(\theta_1 \ldots . \theta_d),$$

$$M_0 = \{\ \}, \qquad M_d = D^{-1}(\{G_j.\theta_{i+1} \ldots . \theta_d \mid 0 \leqslant i \leqslant d \text{ and } G_j \text{ is an atom in } R_i\}.$$

The tuple of sequences $(Sk_d)_0^n, (M_d)_0^n, (\gamma_d)_0^n$ *and* $(Cl_d, j_d)_1^n$ *defines a NMGE.*

As an example consider the successful SLD+abduction derivation in Fig. 4. The sequence of resolvents is

$$\leftarrow p(X, X),\ \leftarrow q(a, V).$$

The sequence of mgu's is

$$\{\ \}\ \{X/a\}.$$

The sequence $(Sk_d)_0^2$ of the dual NMGE is

$$\{\ \}\ \{sk_1\}\ \{sk_1, sk_2\},$$

where $D(sk_1) = X$ and $D(sk_2) = V$. The sequence $(\gamma_d)_0^2$ is

$$\{\ \}\ \{\ \}\ \{sk_1 \rightarrow a\}.$$

The sequence of derived facts of the dual NMGE is

$$\{\ \}\ \{p(sk_1, sk_1)\}\ \{p(a, a), q(a, sk_2)\}.$$

Proof. The lemma can be checked by a straightforward case analysis of the operations that occur during a resolution step and a NMGE computation step. The following correspondences are easily shown.

The selection of the atom in the resolvent and the clause correspond dually to the selection of the rule and the conclusion, respectively. Here we need the fact that each rule in P_D is range-restricted: each universal variable in the rule occurs in the body. If that was not the case, then additional choices had to be made to instantiate the variables occurring in the conclusion only. The duality would be broken.

The renaming of the program clause can be seen as the dual of the skolemisation. This is because the used clauses and the query do not share variables.

The computation of the mgu corresponds to the completion of the set of equalities. This is due to the fact that the completion can be computed incrementally (Lemma 5.10) and the fact that the equation set to be solved for the unification corresponds exactly to the dual of the set of equality atoms to be completed.

The application of the mgu and the addition of the literals of the used clause to the resolvent correspond to the normalisation and assertion phase. □

Theorem 5.11. *For any definite query Q, an abductive refutation for Q and P can be dually interpreted as a successful fair NMGE for only-if$(P)+\neg Q$. The set of atoms of the generated model, restricted to the abducible predicates is the dual of the abductive solution. The dual of the answer substitution is the restriction of γ_n to the skolem constants dual to the variables in the query.*

A failed SLD+abduction derivation corresponds dually to a failed NMGE.

A fair SLD+abduction derivation corresponds dually to a fair NMGE.

A (fair) SLD+abduction tree corresponds dually to a (fair) NMGE tree.

Proof. An SLD+abduction derivation is failed when in the last resolvent an atom of a defined predicate p/n with an empty definition is chosen or else a clause whose head does not unify with the atom. In the first case, the rule $C_{p/n}$ is of the form $\leftarrow p(X_1, \ldots, X_n)$. Hence the NMGE fails. In the second case, the completion of the atoms returns $\{\square\}$ and the NMGE also fails.

In a fair SLD-derivation, each atom occurring in a resolvent is eventually selected. That this implies that the dual NMGE is fair seems evident. Formally, the proof goes as follows. Consider any defined atom $A=p(t_1, \ldots, t_n)$ in LHM_d and the corresponding definition $C_{p/n}=p(\bar{X})\rightarrow E_1, \ldots, E_l$. Define $\sigma_A=\{X_1/t_1, \ldots, X_n/t_n\}$. We show that $LHM_{d'} \models (E_1 \vee \cdots \vee E_l).\sigma_A$ for some $d' \geqslant d$. This is equivalent of showing $LHM_{d'} \models (E_1 \vee \cdots \vee E_l)(\sigma_A.\gamma_{d'})$ (by Proposition 4.8(b)).

Since A occurs in LHM_d, $A.\gamma_d \in M_d$ (Lemma 4.3(c)). Two possibilities exist (by Lemma 5.10): or $A.\gamma_d$ is the normalisation of an atom $B=p(\bar{X}).\sigma_B$ selected for step d or earlier, or $A.\gamma_d$ is the dual of an atom in R_d.

In the first case $LHM_d \models (E_1 \vee \cdots \vee E_l).(\sigma_B.\gamma_d)$ (Lemma 4.16 and Proposition 4.8(a, b)). It suffices to prove that $\sigma_A.\gamma_d \equiv \sigma_B.\gamma_d$. We have that $p(\bar{X}).(\sigma_A.\gamma_d)=A.\gamma_d=B.\gamma_d=p(\bar{X}).(\sigma_B.\gamma_d)$. The identity of $\sigma_A.\gamma_d$ and $\sigma_B.\gamma_d$ follows straightforwardly from this equation and the fact that both substitutions have $\bar{X}$ as domain.

We find that $LHM_d \models (E_1 \vee \cdots \vee E_l)(\sigma_A.\gamma_d)$ and we can take $d'=d$. The second case can be proven in an analogous way using the fairness of the SLD-derivation.

Since the selection in a refutation is fair, a refutation corresponds to a fair successful NMGE.

Because of all previous results, a fair SLD+abduction tree corresponds to a fair NMGET. □

What happens if we drop *FEQ* from *only-if*(P)? In that case, we must replace it by *EQ*. This implies that the completion procedure of *FEQ*, i.e. the dual interpretation of unification must be replaced by a completion procedure of *EQ*, for example, Knuth–Bendix completion. As a consequence the declarative and procedural duality between the model generation and the abduction ceases to exist. Consider the following trivial program P and query Q:

$$r(a)\leftarrow$$

$$\leftarrow r(b).$$

SLD(+abduction) will fail on this query and this corresponds dually to the fact that with FEQ, *only-if*$(P)+\neg Q$ is inconsistent. If we replace FEQ by EQ in *only-if*(P), the set $\{a=b, p(a)\}$ can be extended to a model of *only-if*$(P)+$*not*(Q). This model corresponds to the abductive solution $\{a=b\}$. Most current abductive procedures will not return this solution and this shows that FEQ is inherently present in most current work on abduction in LP.

The following corollary was proved first by Clark [3] for normal programs. For the definite case it follows immediately from the above theorem.

Corollary 5.12. *An SLD refutation for a query Q and a definite program P without abducibles is a consistency proof of only-if$(P)+\neg Q$. A failed SLD tree for a ground query Q and P is an inconsistency proof of only-if$(P)+\neg Q$, and therefore of comp$(P)+\neg Q$.*

Theorem 5.11 indicates that in a fair SLD+abduction tree, all branches correspond dually to fair NMGEs, and hence dually generate models. However, only the finite branches generate abductive solutions. So, in the case of an infinite branch, the duality is broken. Here is a trivial example of this situation. Consider the definite program $P=\{p\leftarrow p\}$ and the query $\leftarrow p$. No SLD refutation for the query exists and the SLD tree consists of one infinite branch $p\leftarrow p\leftarrow p\leftarrow\cdots$ *only-if*$(P)+\neg Q$ is the theory $\{p\leftarrow\ p\leftarrow p\}$. The dual of the infinite SLD derivation is a fair NMGE and generates the model $\{p\}$.

What this example shows is that infinite fair NMGEs generate models which do not correspond to abductive solutions. Do finite NMGEs always generate models corresponding to abductive solutions? Unfortunately, this is not the case either. Consider the following NMGE for the same theory as in the previous paragraph: $M_0=\phi$; $M_1=\{p\}$. This is a finite fair NMGE, but the set of abductive atoms in the model $(=\phi)$ is not an abductive solution.

There is an important class of definite abductive programs where the duality is perfect, namely for definite abductive *acyclic programs* and *bounded queries* [1]. For these programs and queries, and SLD+abduction tree is always finite. Using this fact and the completeness Theorem 4.15, it is easy to prove that the abductive atoms in each model of the dual theory form an abductive solution.

6. Implementing NMGE

We have implemented two instances of the NMGE framework, one for FEQ and one for EQ. The model generator for FEQ is easy to implement, since E-unification can be replaced by unification (Proposition 5.7) and FEQ has a simple, incremental completion procedure. Two technical problems deserve special attention. One problem is that all universal variables of a rule being applied must be instantiated with

ground terms and the procedure matching bodies of rules with elements of M_d only instantiates variables of the body. We circumvent this problem by requiring that the rules are in *range-restricted form* (i.e. all universal variables occurring in the head occur in the body in a nonequality atom), and by transforming each theory violating this condition to range-restricted form. This can be done by introducing a domain predicate $\mathscr{U}/1$ representing the domain of interpretation. For each universal variable X not occurring in the body, $\mathscr{U}(X)$ is added to the body. For each existential variable X in a conclusion, $\mathscr{U}(X)$ is added to the conclusion. In addition, rules of the form $\mathscr{U}(X_1), \ldots, \mathscr{U}(X_n) \rightarrow \mathscr{U}(f(X_1, \ldots, X_n))$ are added for each functor f/n $(n \geqslant 0)$.

A second problem is related to the fairness condition. Theorem 4.15 proves that a fair NMGET exists, but without clarifying how to implement the condition. The solution that we have adopted is the one used in Satchmo [18]: *level saturation*. The idea is to generate conclusions level by level. For a given level with associated M_d, γ_d, normal instances of rules which are violated in LHM_d are selected, conclusions are selected, skolemisation is performed but all facts in the selected conclusion are stored apart. Only when all violated instances have been applied, the completion γ_{d+1} of γ_d and the derived equality facts is computed, the derived nonequality facts are added to M_d and normalisation is applied, yielding M_{d+1}.

A second instance that was implemented is for EQ as underlying equality theory. The completion of a *ground* TRS can be computed [9, 25] and moreover, efficient algorithms exist [25]. Hence, EQ is an equality theory with completion. Our prototype uses *narrowing* [19] to compute normal E-unifiers, and an optimised form of the Knuth–Bendix algorithm [15] as completion procedure. The model generator operates on range-restricted programs. The fairness condition is implemented using *level saturation*.

Experiments with both systems are promising. They show that the dynamic contraction, implemented by dynamic completion and normalisation, can avoid exponential explosion and even looping due to the equality axioms.

7. Discussion

A current limitation of the duality framework is its restriction to definite abductive programs. In the future we will extend it to the case of normal abductive procedures. The extended framework will then describe a duality between an SLDNF + abduction procedure and a form of model generation.

The SLDNF + abduction procedure can be found by proceeding as for the definite case. There we started from pure SLD and definite programs without abduction, we dualised it and obtained the NMGE method, which under dualisation yields an SLD + abduction procedure. At present we have performed (on an informal basis) the dualisation of SLDNF for normal programs without abduction. Under dualisation, the resulting model generation procedure gives a natural extension of SLDNF for

abductive programs. The abductive procedure incorporates skolemisation for non-ground abducibles goals and efficient treatment of abduced equality atoms by the methods presented earlier. Integrity constraints can be represented by adding, for any integrity constraint IC, the rule: "$false \leftarrow not(IC)$.", transforming these rules to a normal program using the transformation of Lloyd–Topor [17], and adding the literal *not false* to the query.

A prototype of this method has been implemented. An interesting experiment was its extension to an abductive planner based on the event calculus. Our prototype planner was able to solve some hard problems with context-dependent events, problems that are not properly solved by existing systems [23, 22].

In [7, 8], we proved the soundness of the procedure with respect to completion semantics, in the sense that for any query $\leftarrow Q$ and generated solution Δ:

$$P + \Delta \models Q.$$

This implies the soundness of the procedure with respect to the generalised stable model semantics of [14]: a generated solution can be extended in a natural way to a generalised stable model of the abductive program. As a completeness result we proved that the procedure generates all *minimal* solutions when the computation tree is finite.

Related to our work, [2] also indicates a relationship between abduction and model generation. The nature of this work differs from ours. The goal of [2] is to develop an abductive procedure in the context of updating deductive databases. A metaprogram is proposed which takes a query and an abductive program P as input and, when executed by a model generator, generates abductive solutions. Our work takes the alternative approach of executing the model generator directly on *only-if*(P). This allows us to present a more explicit duality, not only on the level of the abductive solutions and the generated models, but also on the fine-grained computational steps involved in the applied abduction and model generation procedures. The meta-approach of [2] makes no reference to the only-if part of the abductive program. In addition, on a more technical level, no equality atoms appear in the head of the metaprogram and, therefore, no special treatment for FEQ is necessary.

In [4], another approach is taken for abduction through deduction. An abductive procedure is presented which, for a given normal abductive program P and query $\leftarrow Q$, derives an *explanation formula* E equivalent to Q under the completion of P:

$$comp(P) \models (Q \Leftrightarrow E).$$

The explanation formula is built from abducibles predicates and equality only. It characterises all abductive solutions in the sense that, for any set Δ of abducible atoms, Δ is an abductive solution iff it satisfies E.

Although this approach departs also from the concept of completion, it is of a totally different nature. In the first place, our approach aims at contributing to the procedural semantics of abduction. This is not the case with the work in [4]. Another difference is that this approach is restricted to queries with a finite computation tree. If

the computation tree contains an infinite branch, then the explanation formula cannot be computed.

In [13], an abductive procedure for normal abductive programs has been defined. A restriction of this method is that abducible goals can only be selected when they are ground. This poses a serious problem for applications such as planning. The methods presented here allow to overcome the problem by skolemisation of nonground goals and efficient treatment of abduced equality facts. As argued in Section 1, in planning, the skolemisation of nonground atoms is used to introduce new events which initiate some desired goal.

Recently, a planning system based on abduction in the event calculus has been proposed in [22]. The underlying abductive system incorporates negation as failure, skolemisation for nonground abducible goals and efficient treatment of abduced equality facts. However, the system shows some problems with respect to soundness and completeness. Experiments indicated that these problems are solved by our prototype planner.

Finally, we want to draw attention to an unexpected application of the duality framework. An uncommon form of abduction is obtained if FEQ is replaced by general equality EQ and the equality predicate is abducible. This form of abduction is presented in [5]. Take the program $P=\{r(a)\leftarrow\}$. For this program, the query $\leftarrow r(b)$ has a successful abductive derivation

$$\leftarrow \underline{r(b)} \quad \Delta=\{\ \},$$
$$\square \qquad \Delta=\{b=a\}.$$

$\leftarrow r(b)$ succeeds under the abductive hypothesis $\{b=a\}$. The duality framework provides the technical support for efficiently implementing this form of abduction. The difference with normal abduction is that the completion procedure for FEQ – the dual of unification – must be replaced by a completion procedure for EQ, for example, Knuth–Bendix completion.

To conclude, we have presented a duality between two computational paradigms. This duality allows to transfer technical results from one paradigm to the other and vice versa. One application that was obtained was an efficient extension of model generation with equality. Transferring these methods back to abduction, we obtained techniques for dealing with nonground abducible goals and efficient treatment of abduced equality atoms. We discussed experiments indicating that the extension of the duality framework for the case of normal programs is useful for obtaining an abductive procedure for normal abductive programs.

Acknowledgment

We thank Krzysztof Apt, Eddy Bevers, Maurice Bruynooghe and Francois Bry for helpful suggestions.

Appendix. Expressive power of the extended clause formalism

We prove that for each first-order logic theory T based on $\mathscr{L}$ there exists an equivalent theory T', consisting of extended clauses and based on a language $\mathscr{L}'$, which extends $\mathscr{L}$ by a finite set of predicate symbols. Here equivalence means that each model of $\langle \mathscr{L}, T \rangle$ can be extended to a model of $\langle \mathscr{L}', T' \rangle$ and vice versa, that the restriction of a model of $\langle \mathscr{L}', T' \rangle$ to the symbols of $\mathscr{L}$ is a model of T. This implies that $\langle \mathscr{L}', T' \rangle$ is a *conservative extension* of $\langle \mathscr{L}, T \rangle$ [24]: for each formula F based on $\mathscr{L}$, $\langle \mathscr{L}, T \rangle \models F$ iff $\langle \mathscr{L}', T' \rangle \models F$. This form of equivalence is stronger than the form of equivalence which has been proven for a theory T and its clausal form T': T is consistent iff T' is consistent. This result does not guarantee that, for any formula F based on the language of T, $T \models F$ iff $T' \models F$. On the other hand, if, for any formula F based on the language of T, $T \models F$ iff $T' \models F$, then it follows that T is consistent iff T' is consistent.

We use the following terminology. $\bar{X}$ and $\bar{Y}$ denote tuples of variables $(X_1, \ldots, X_n)$ and $(Y_1, \ldots, Y_m)$. The notation $F[\bar{X}]$ is used for a formula F to denote that $X_1, \ldots, X_n$ are the free variables of F. We denote the fact that F_c is a subformula of F by $F_c \leqslant F$, and that F_c is a strict subformula of F by $F_c < F$. The set of *components* of a formula F are defined as the set of maximal strict subformulas of F. A conjunction and disjunction have two components; negations, universal and existential formulas have one component. For each subformula F_c in F, there exists a linear chain of formulas $F_c = F_0 < F_1 < \cdots < F_n = F$, where each F_i is a component of F_{i+1} (although our notations do not make this explicit, we are talking about occurrences of subformulas rather than of subformulas directly; this is to avoid problems in the case of a subformula with multiple occurrences). The set $\{F_1, \ldots, F_n\}$ is precisely the set of formulas F' such that $F_c < F' \leqslant F$. The depth of F_c in F is n. The depth of a formula is recursively defined as the maximum depth of its components augmented with one. F_c occurs in a positive context or occurs positively in F if the number of formulas G such that $F_c < \neg G \leqslant F$ is even. Otherwise, F_c occurs in a negative context or occurs negatively in F.

We assume that in a closed formula F each variable occurs with precisely one quantifier. When this is not the case, renaming is always possible. Further, we require that each formula contains only the connectors $\wedge$, $\vee$ and $\neg$ and, moreover, that each negation in the formula has an atom as component. It is well known that each first-order logic formula can be transformed to an equivalent formula which satisfies these conditions.

In Section 2, the notion of extended clause was defined. The lemma below gives another characterisation.

Lemma A.1. *A formula F is an extended clause iff it satisfies the following constraints:*

- *the component of a negation $G \leqslant F$ is an atom (as required by syntactic constraints given earlier);*
- *the components of a conjunction $G \leqslant F$ are atoms or conjunctions;*

- *the component of an existential formula $G \leqslant F$ is either an atom or a conjunction or an existential formula;*
- *the components of a disjunction $G \leqslant F$ are atoms, negations, conjunctions, existential formulas or disjunctions;*
- *a universal formula $G \leqslant F$ can have any type of component.*

Proof. A sketch of the proof is given. An extended clause is a formula of the form

$$\forall X_1 \cdots \forall X_f \colon \neg A_1 \vee \cdots \neg A_g \vee E_1 \vee \cdots \vee E_h,$$

where E_i has the general form

$$\exists Y_1, \ldots, Y_i \colon A_1 \wedge \cdots \wedge A_j.$$

Each A_i denotes an atom and $f, g, h, i \geqslant 0$; $j > 0$ and $g > 0$ or $h > 0$.

It is straightforward that an extended clause satisfies the syntactical constraints of the lemma. Conversely, assume that F satisfies the syntactic constraints of the lemma. A proof by induction on the depth of the formula F can be given. The idea is as follows: by the induction hypothesis the components of F are known to be extended clauses. Since F satisfies the constraints in the lemma, the type of F restricts the type of its components. A simple case analysis suffices to show that F must be an extended clause too. For example, let F be a disjunction $G_1 \vee G_2$. G_1, G_2 can be any formula except a universal formula and they are both extended clauses. Hence, they are both of the form $\neg A_1 \vee \cdots \vee \neg A_g \vee E_1 \vee \cdots \vee E_h$. The disjunction of two formulas of this form is again of this form (strictly spoken, commutativity and associativity must be applied here). Hence, F is an extended clause. □

Below, an algorithm is given which transforms any theory to an extended program. The transformation proceeds by iteratively replacing an unwanted subformula of a formula (i.e. a subformula of a type which is not allowed by the lemma) by an atom of a new predicate, and adding a formula which relates this new predicate and the replaced subformula.

Algorithm A.2. *Let T be a theory based on $\mathscr{L}$. The transformation algorithm is defined as follows. Initially, set $T' = T$. As long as T' contains unwanted formulas, the following transformation step is executed:*

- *Select F from T' such that F contains a subformula G with an unwanted component $F_c[\bar{X}]$.*
- *Choose a new predicate p/n.*
- *Define F' by replacing F_c in F by $p(\bar{X})$.*
- *If F_c is a universal formula then F_c is of the form $\forall \bar{Y} \colon F'_c$ with F'_c not a universal formula; define $F'' = \forall \bar{X}, \bar{Y} \colon \neg p(\bar{X}) \vee F'_c$.*
  Otherwise, define $F'' = \forall \bar{X} \colon \neg p(\bar{X}) \vee F_c$.
- *Define $T' := T \setminus \{F\} \cup \{F', F''\}$.*

Theorem A.3. *Let T be any first-order theory. The transformation algorithm terminates always. It produces an extended program T' based on an extension $\mathscr{L}'$ of $\mathscr{L}$ with new predicate symbols. The following equivalence holds between T and T':*

(a) *Each model of $\langle \mathscr{L}, T \rangle$ can be extended to a model of $\langle \mathscr{L}', T' \rangle$.*

(b) *The restriction of a model of $\langle \mathscr{L}', T' \rangle$ to the symbols of $\mathscr{L}$ is a model of $\langle \mathscr{L}, T \rangle$.*

(c) *For each sentence F based on $\mathscr{L}$: $\langle \mathscr{L}, T \rangle \models F \Leftrightarrow \langle \mathscr{L}', T' \rangle \models F$.*

Proof. To see that the algorithm terminates, just check that each transformation step decreases the number of unwanted components with one.

That the resulting T' is an extended program is trivial since no formula in T' contains an unwanted component.

For the proofs of (a) and (b), we simplify the situation: we first show that if T' is obtained by T by one transformation step then (a) and (b) hold. Then, by induction, (a) and (b) hold for the complete transformation process.

Assume that the transformation step selects F from T'. F contains a subformula with an unwanted component F_c with free variables $\bar{X}$. To prove (a), let M be a model of $\langle \mathscr{L}, T \rangle$. We extend M to M' by extending the truth function of M' in the following way: for each variable assignment V of the variables $\bar{X}$: $M', V \models p(\bar{X})$ iff $M, V \models F_c$.

By definition, $M' \models \forall \bar{X}: (p(\bar{X}) \leftrightarrow F_c)$. From this equivalence, it is easy to prove that $M' \models F' \wedge F''$. In the case that F_c is a universal formula, the proof relies on the equivalence

$$(\forall \bar{X}: (p(\bar{X}) \rightarrow \forall \bar{Y}: F'_c)) \Leftrightarrow (\forall \bar{X}, \bar{Y}: p(\bar{X}) \rightarrow F'_c),$$

which holds because $\bar{X}$ and $\bar{Y}$ are disjunct.

To prove (b), we use Lemma A.4 which is formulated and proven below: assume that F contains a subformula $F_c[\bar{X}]$ in a positive context. Let F' be obtained by replacing F_c by F'_c, a formula with the same free variables. Then $(\forall \bar{X}: (F'_c \rightarrow F_c)) \rightarrow (F' \rightarrow F)$ is a tautology.

Since $M' \models \forall \bar{X}: (p(X_1, \ldots, X_k) \rightarrow F_c)$ and $M' \models F'$, if follows that $M' \models F$.
The proof of (c) follows directly from (a) and (b). □

The above transformation procedure is far from optimal in the sense that it often introduces a large number of new predicates. In general, a better result will be obtained if T is first pre-processed by distributing existential quantifiers over disjunctions and universal quantifiers over conjunctions.

Lemma A.4. *Let $F[\bar{Y}]$ be a formula containing a subformula $F_c[\bar{X}]$. Let F' be the formula obtained by replacing F_c by F'_c. If F_c occurs in a positive context in F then $\forall \bar{X}: (F_c \rightarrow F'_c)$ implies $\forall \bar{Y}: (F \rightarrow F')$. If F_c occurs in a negative context then $\forall \bar{X}: (F_c \rightarrow F'_c)$ implies $\forall \bar{Y}: (F' \rightarrow F)$.*

Proof. The proof is by induction on the depth of F_c w.r.t. F. The induction step is based on the following implications which can easily be proved, for example, in a model-theoretic way:

$$\forall X_1, \ldots, X_n\colon \quad (F \rightarrow F') \Rightarrow \forall X_1, \ldots, X_{n-1}\colon ((\forall X_n\colon F) \rightarrow (\forall X_n\colon F'))$$

$$\forall X_1, \ldots, X_{n-1}\colon ((\exists X_n\colon F) \rightarrow (\exists X_n\colon F'))$$

$$\forall X_1, \ldots, X_n\colon (F \wedge G \rightarrow F' \wedge G)$$

$$\forall X_1, \ldots, X_n\colon (F \vee G \rightarrow F' \vee G)$$

$$\forall X_1, \ldots, X_n\colon (\neg F \leftarrow \neg F').$$

The case that F_c occurs at depth 0 in F (i.e. $F_c = F$, $F'_c = F'$) is trivial.

Assume that the lemma is proved for formulas in which F_c occurs at depth d. We prove the lemma for a formula F in which F_c occurs at depth $d+1$. F is either of the form $\exists X\colon F_1$, $\forall X\colon F_1$, $F_1 \wedge F_2$, $F_1 \vee F_2$ or $\neg F_1$.

If F_c occurs positively (negatively) at depth $d+1$ in F and F is not a negation $\neg F_1$ then F_c occurs positively (negatively) in F_i ($i=1$ or $i=2$) at depth d. By the induction hypothesis it holds that $\forall Y_1, \ldots, Y_k\colon F_i \rightarrow F'_i$ (or $\forall Y_1, \ldots, Y_k\colon F_i \leftarrow F'_i$ if F_c occurs negatively in F_i). Here F'_i is the formula obtained by substituting F_c for F'_c in F_i. From the above implications, it follows directly that $\forall \bar{Y}\colon F \rightarrow F'$.

When F_c occurs negatively (positively) in $\neg F_1$, then F_c occurs positively (negatively) in F_1. Because of the induction hypothesis, it holds that $\forall Y_1, \ldots, Y_k\colon F_1 \rightarrow F'_1$ (or $\forall Y_1, \ldots, Y_k\colon F_1 \leftarrow F'_1$ if F_c occurs negatively in F_1). Because of the implication for the negation, the implication switches. So the lemma holds. □

References

[1] K.R. Apt and M. Bezem, Acyclic programs, in: *Proc. 17th Internat. Conf. on Logic Programming* (MIT Press, Cambridge, MA, 1990) 579–597.

[2] F. Bry, Intensional updates: abduction via deduction, in: *Proc. Internat. Conf. on Logic Programming '90* (1990) 561–575.

[3] K.L. Clark, Negation as failure, in: H. Gallaire and J. Minker, eds., *Logic and Databases* (Plenum, New York, 1978) 293–322.

[4] L. Console, D. Theseider Dupre and P. Torasso, On the relationship between abduction and deduction, *J. Logic Computation* **1** (1991) 661–690.

[5] P.T. Cox, E. Knill and T. Pietrzykowski, Abduction in logic programming with equality, in: *Proc. Internat. Conf. on Fifth Generation Computer Systems 1992* (1992) 539–545.

[6] P.T. Cox and T. Pietrzykowski, Causes for events: their computation and application, in: *Proc. 8th Internat. Conf. on Automated Deduction*, 1986.

[7] M. Denecker and D. De Schreye, A family of abductive procedures for normal abductive programs, their soundness and completeness, Tech. Report 136, Department of Computer Science, K.U. Leuven, 1992.

[8] M. Denecker and D. De Schreye, SLDNFA; an abductive procedure for normal abductive programs, in: K. Apt, ed., *Proc. Internat. Joint Conf. and Symp. on Logic Programming*, Washington 1992.

[9] N. Dershowitz, Completion and its applications, in: *Proc. CREAS*, 1987.

[10] N. Dershowitz and J.-P. Jouannaud, Rewrite systems, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science, Vol. B*, Ch. 15 (North-Holland, Amsterdam, 1989).
[11] K. Eshghi, Abductive planning with event calculus, in: R.A. Kowalski and K.A. Bowen, eds., *Proc. 5th ICLP*, 1988.
[12] G. Huet, Confluent reductions: abstract properties and applications to term rewriting systems, *J. ACM* **27** (1980) 797–821.
[13] A.C. Kakas and P. Mancarella, Database updates through abduction, in: *Proc. 16th Very Large Database Conf.* (1990) 650–661.
[14] A.C. Kakas and P. Mancarella, Generalised stable models: a semantics for abduction, in: *Proc. ECAI-90*, 1990.
[15] D.E. Knuth and P.B. Bendix, Simple word problems in universal algebras, in: J. Leech, ed., *Computational Problems in Abstract Algebra* (Pergamon, Oxford, 1970) 263–297.
[16] R.A. Kowalski, Logic programming in artificial intelligence, in: *Proc. IJCAI*, 1991.
[17] J.W. Lloyd and R.W. Topor, Making prolog more expressive, *J. Logic Programming* **1** (1984) 225–240.
[18] R. Manthey and F. Bry, A hyperresolution-based proof procedure and its implementation in prolog, in: *Proc. 11th German Workshop on Artificial Intelligence* (Geseke, 1987) 221–230.
[19] A. Martelli, C. Moiso and C.F. Rossi, An algorithm for unification in equational theories, in: *Proc. Symp. on Logic Programming* (1986) 180–186.
[20] A. Martelli and U. Montanari, An efficient unification algorithm, *Trans. Programming Languages Systems* **4** (1982) 258–282.
[21] Y. Metivier, About the rewriting systems produced by the Knuth–Bendix completion algorithm, *Inform. Process. Lett.* **16** (1983) 31–34.
[22] L. Missiaen, Localized abductive planning with the event calculus, Ph.D. Thesis, Department of Computer Science, K.U. Leuven, 1991.
[23] M. Shanahan, Prediction is deduction but explanation is abduction, in: *Proc. IJCAI '89* (1989) 1055.
[24] J. Shoenfield, *Mathematical Logic* (Addison-Wesley, Reading, MA, 1967).
[25] W. Snyder, Efficient ground completion: an $o(n \log n)$ algorithm for generating reduced sets of ground rewrite rules equivalent to a set of ground equations, in: *Proc. 3rd Internat. Conf. on Rewriting Techniques and Applications*, 1989.
[26] M. van Emden and R.A. Kowalski, The semantics of predicate logic as a programming language, *J. ACM* **4** (1976) 733–742.